(2)

THE IAI EXP[illegible]THE USE OF ECSAM[illegible]THE[illegible]EMENT®

[illegible]NT

[illegible]C-9[illegible]/-CMC

[illegible]NO[illegible]1993

Software Productivity Consortium

THE IAI EXPERIENCE IN THE USE OF ECSAM IN THE STATEMENT® ENVIRONMENT

SPC-93127-CMC

VERSION 01.00.00

NOVEMBER 1993

JONAH Z. LAVI

This material is based in part upon work sponsored by the Advanced Research Projects Agency under Grant # MDA972-92-J-1018. The content does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

This document accompanies a videotape of the same presentation recorded live at the Software Productivity Consortium in August 1993. It is recommended that the videotape be viewed with these viewgraphs at hand.

Produced by the
SOFTWARE PRODUCTIVITY CONSORTIUM

under contract to the
VIRGINIA CENTER OF EXCELLENCE
FOR SOFTWARE REUSE AND TECHNOLOGY TRANSFER
SPC Building
2214 Rock Hill Road
Herndon, Virginia 22070

DR. JONAH Z. LAVI

CBSE ASSOCIATES

THE IAI EXPERIENCE IN THE USE OF ECSAM IN THE STATEMATE® ENVIRONMENT

ABSTRACT

ECSAM is a comprehensive object based model driven method for the analysis of Computer Based Systems (CBS) and their software. It is being developed at the Israel Aircraft Industries (IAI) for the analysis and design of complex embedded computer systems and their software. Using this method, systems are described by two complementary models: a conceptual model and a design model. The conceptual model describes a system in terms of its conceptual structure and interfaces, its capabilities, and its dynamic behavior. It is represented by three interrelated partially overlapping views: the logical modules view, the operating modes view, and the dynamic processes view. The design model describes the systems in terms of its architecture, hardware, and software subsystems, communication lines and their properties, the human-machine interface design and implementation constraints. The ECSAM model is presented during the lecture and demonstrated using an example. Three application examples are discussed and lessons learned during their application are presented.

The IAI Experience in the Use of ECSAM in the STATEMATE™ Environment

Jonah Z. Lavi - CBSE Associates

August 1993

Topics

- Lecture objective
- The IAI approach to ECS development
- Typical application of ECSAM and STATEMATE in IAI projects
- Overview of the ECSAM method
- Development history highlights
- CBSE Histories
- Summary

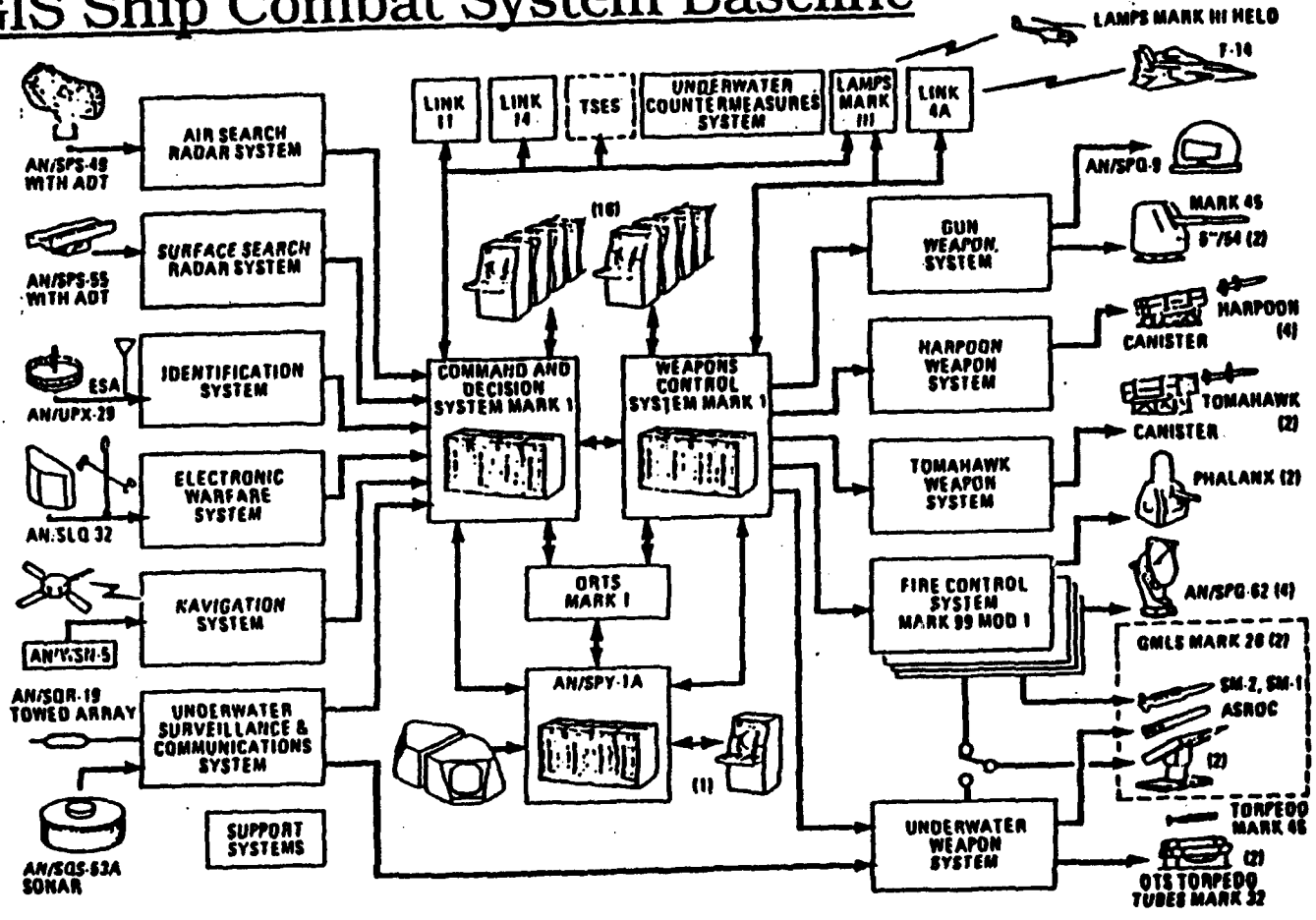
Objective

To present project experience and lessons learned using IAI's ECSAM method supported by the STATEMATE environment in the development of CBS

Modern ECS Characteristics

- Very complex
- Multi-systems
- Multi-level
- Multi-computer
- Complex communications
- Contain hardware and software in each subsystem

AEGIS Ship Combat System Baseline



A typical Large System

The IAI Approach

- The basic principle:

  Model driven systems and software development

- All requirement analysis and design activities are aimed at creating a clear, consistent, and testable model of the system
- Document generation is a "window" into the model, not a process driver
- Concise evolving documents, based on graphics, are used during most project phases

Typical Application of ECSAM and STATEMATE in IAI Projects

- The ECSAM / STATEMATE environment is presently applied in IAI projects to create:
  - Top-level system and subsystem ("SEGMENT") requirements
  - Computer-based systems designs
  - Software requirements
  - Software interface specifications
  - Software interface designs
  - Top-level software designs
- Generation of documents according to IAI "Evolving Specs" or DoD-STD-2167A formats for the above activities

Typical Application of ECSAM and STATEMATE in IAI Projects (cont.)

- DoD-STD-2167A documents generated with templates developed at IAI; approved by the DoD and other customers
- IAI prototype tool used for allocation of requirements from contract documents to ECSAM model in a STATEMATE database
- Traceability monitoring and documentation of the allocated requirements
- ECSAM / STATEMATE used to define, simulate, and analyze complex system dynamics and their software implementation

Typical Application of ECSAM and STATEMATE in IAI Projects (cont.)

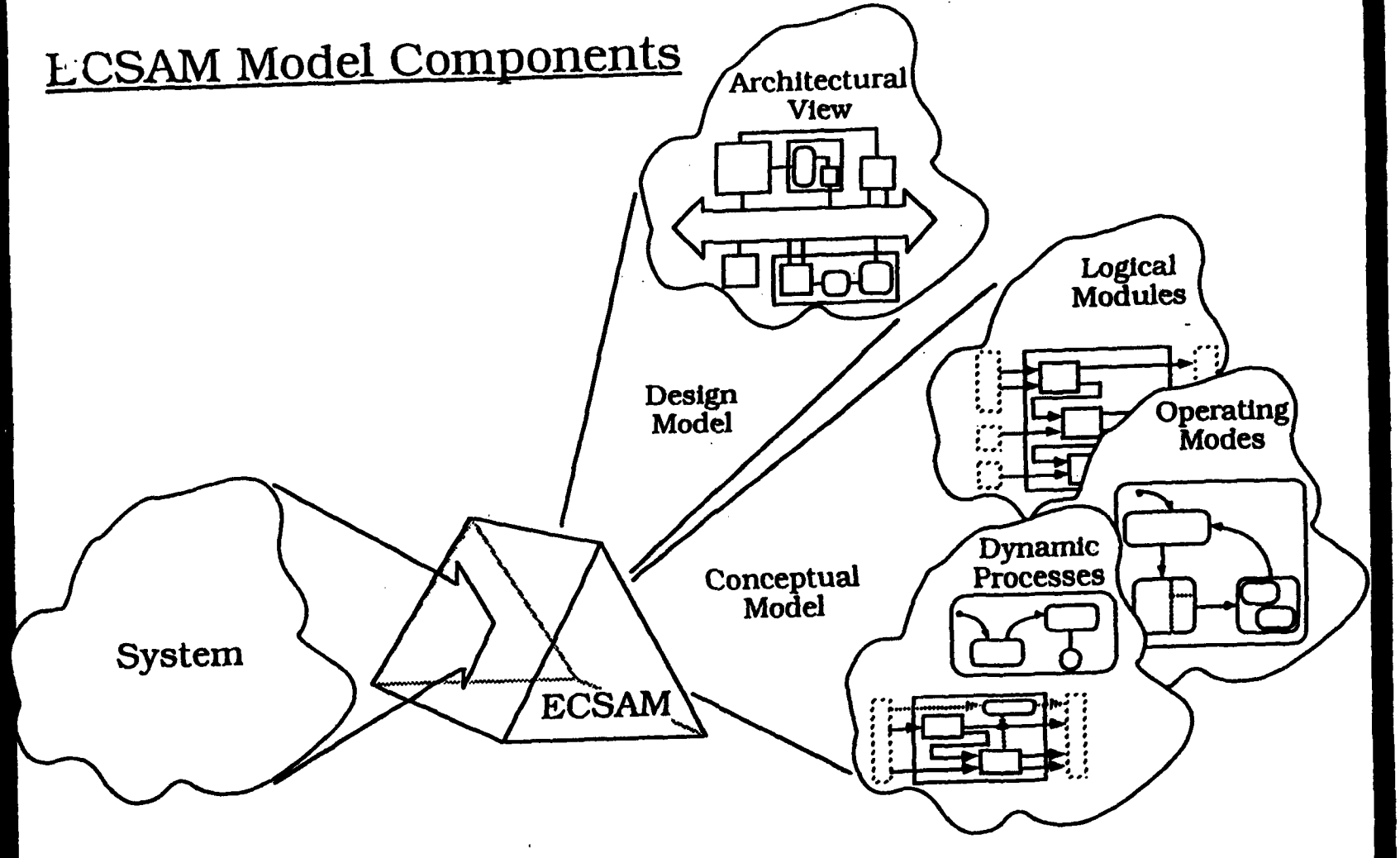
- Current applications include:
  - Integrated Unmanned Air Vehicle (UAV) systems
  - Communication satellites
  - Avionic systems development and upgrades
  - Optronic systems and payloads
  - . . .

Overview of the ECSAM Method

ECSAM

A model-based method for the analysis and modeling of Embedded Computer Systems (ECS) and their software

ECSAM Model Components



ECSAM Method Background (1)

- The ECSAM method has been developed since 1980 at IAI while working on real projects
- The following ECSAM components where developed at IAI:
  - The ECS generic model
  - STATECHARTS
  - The ECSAM model, consisting of:
    - The conceptual model
    - The design model
  - The evolving documentation concept

ECSAM Method Background (2)

- ECSAM incorporates concepts from known systems and software engineering methods. Among those are:
  - Object-based modeling
  - "Parnas Methods"
  - SADT
  - RT-SD
  - Structured Programming

The ECSAM Models (1)

- The Conceptual Model

  Is described by the following views:

  - The Logical Modules View

    Describes the system's partitioning into logical subsystems (objects), the functional capabilities (services) performed by them, and data/signal flows between them

  - The Operating Modes View

    Describes the main operating modes of the system and the transitions between them

  - The Dynamic Process View

    Describes the behavioral processes in the system, occurring in response to external/internal events

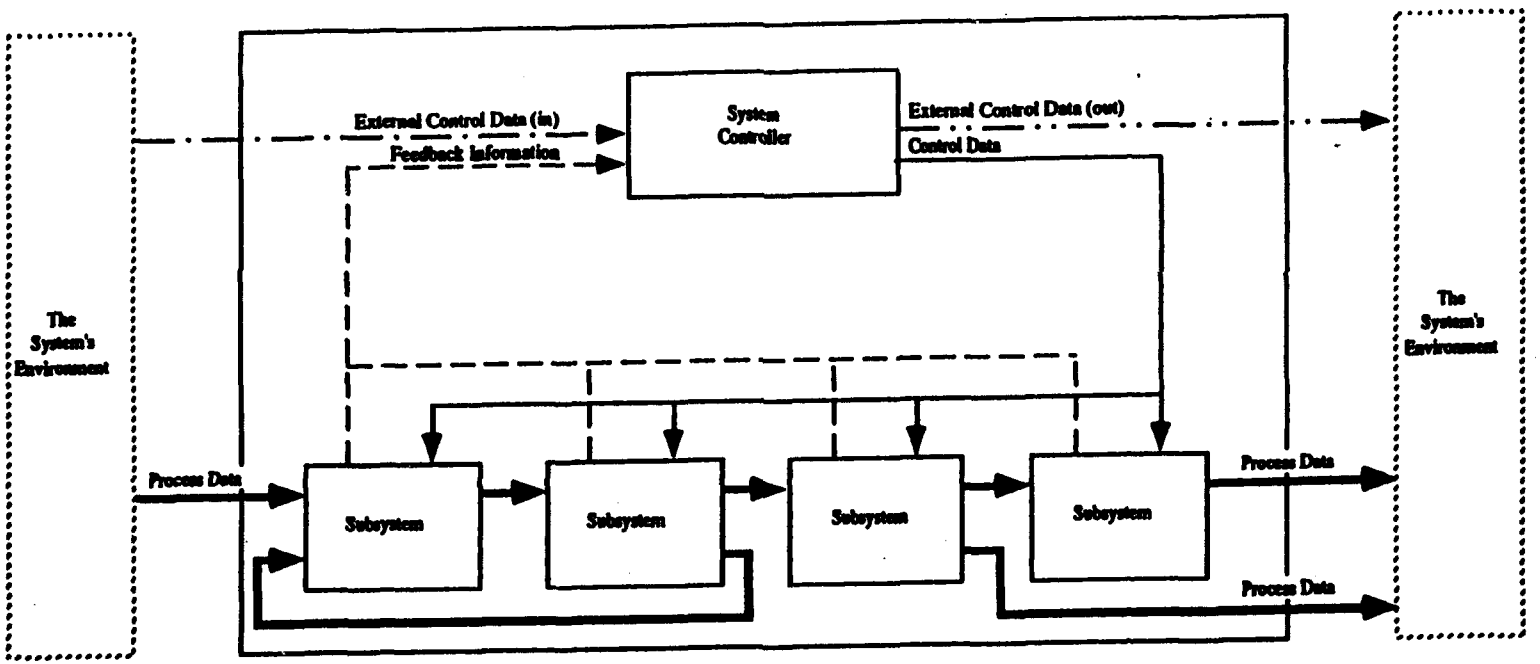
The ECSAM Models (2)

- The Design Model

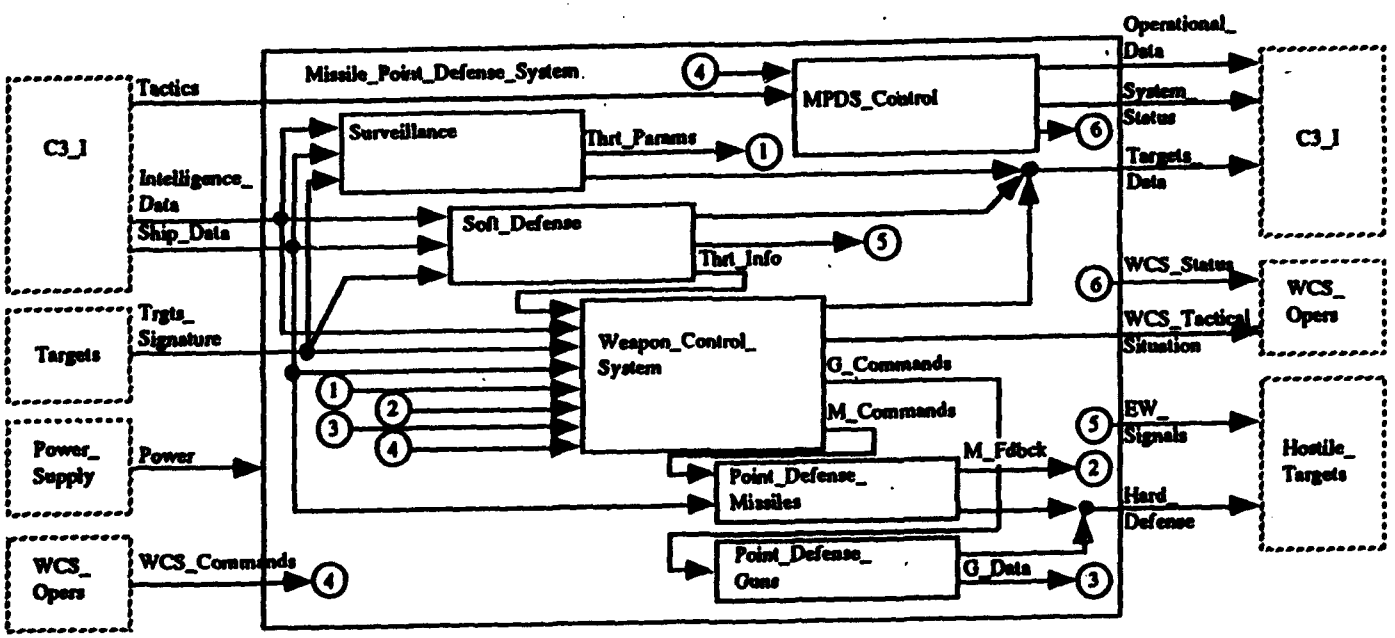
  Describes the system in terms of its:

  - Architectural view
  - Hardware and software subsystems and their interaction
  - Communication lines and their properties
  - Human - machine interface design and implementation constraints

ECSAM Computer-Based System Generic Model



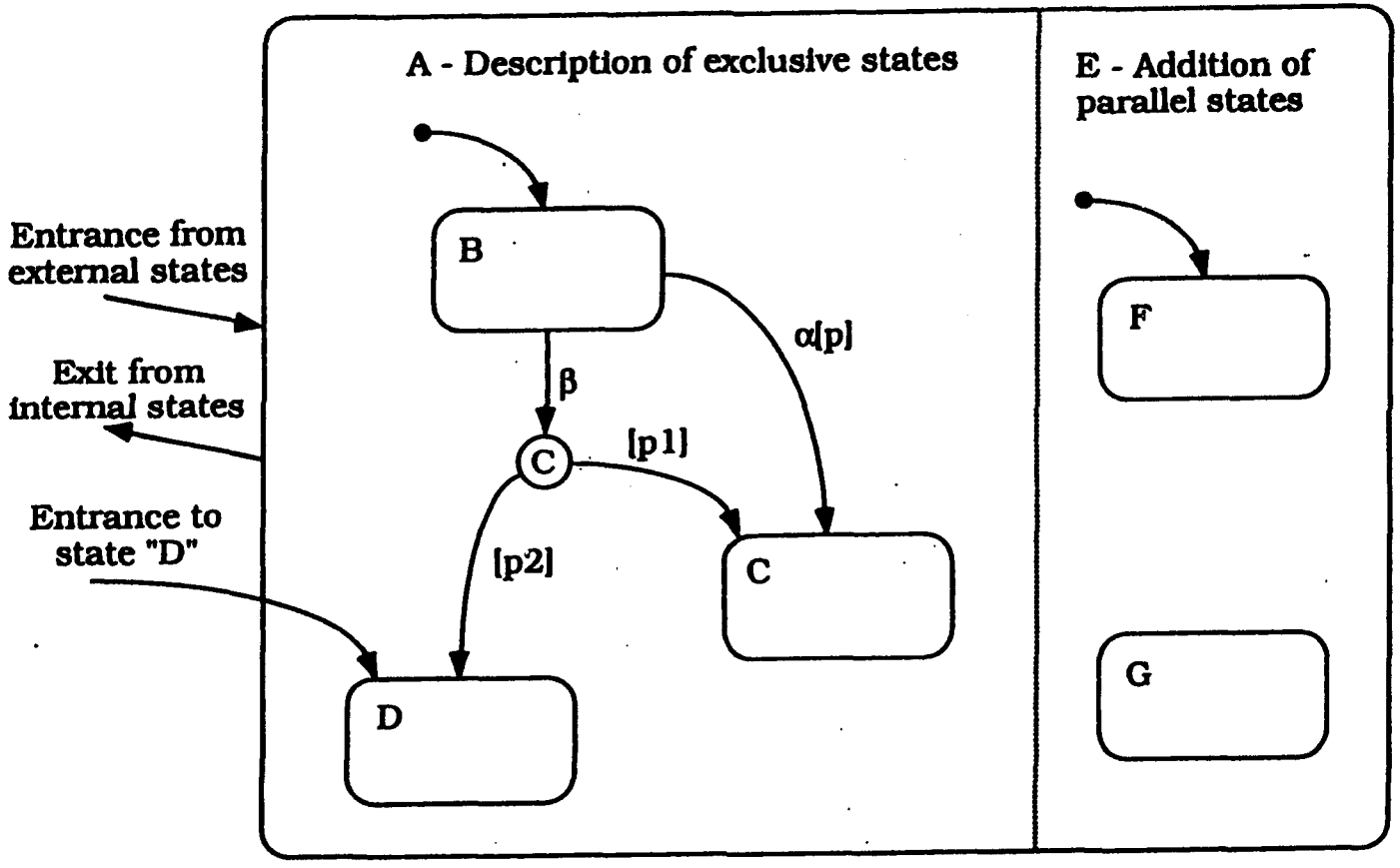
MPDS Logical Model



MPDS Subsystems Sample Capabilities

Surveillance Radar (SR)	Soft Defense (EW)	Weapon System (WCS)	Point Defense Missiles (PDM)	Point Defense Guns (PDG)
• Detect_target • Identify_ target • Create_ target_track • Update_ target_track • . . .	• Deceive_ agressor • Check_ deception_ success • . . .	• Position_FCS • Acquire_ lock_target • Track_target • Solve_gdnc_ equations • Pre_launch_ check • Evaluate_hit • Guide_missile • . . .	• Prepare_ missiles • Pre_launch_ check • Launch • . . .	• Prepare_guns • Control_aim • Fire_guns • . . .

STATECHARTS



Example of Transition Specification from the A-7 Project

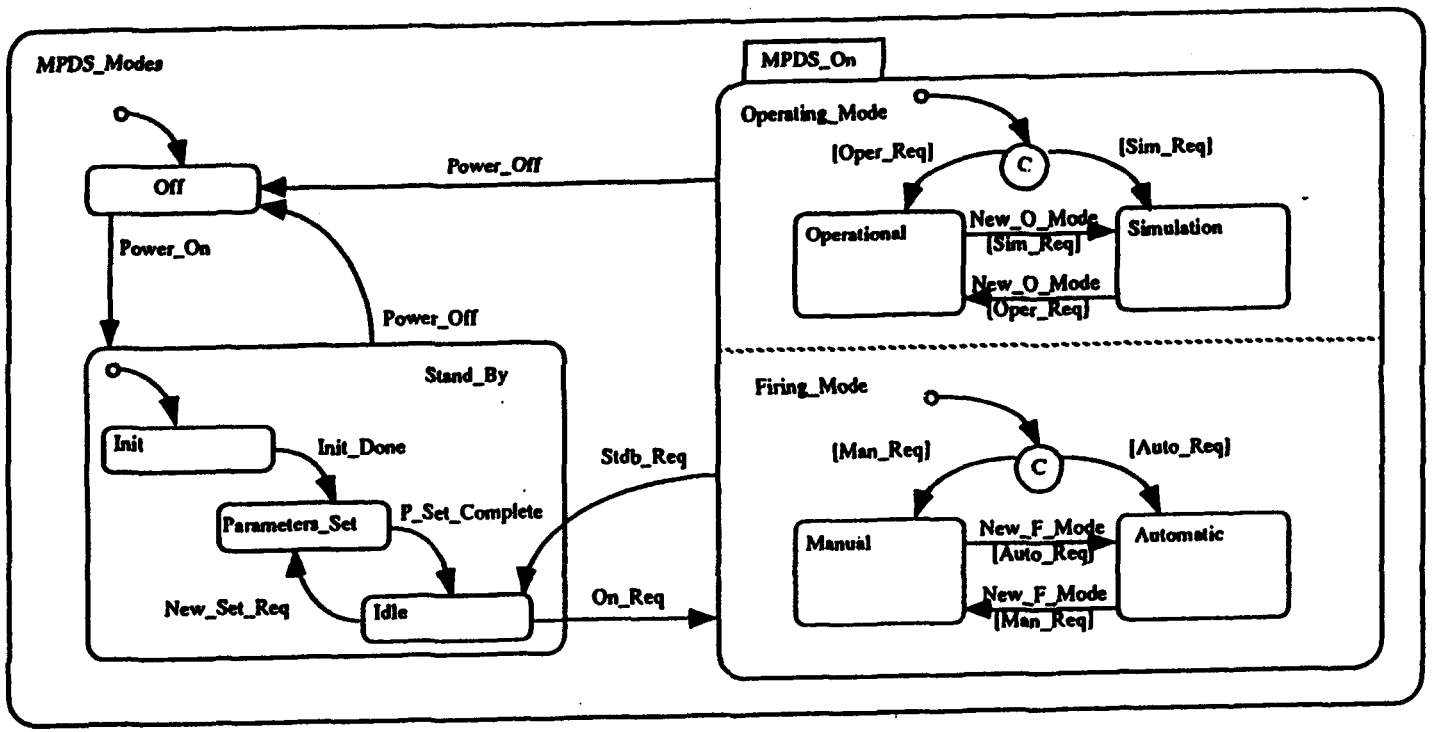
HUDaln to *DIG*

@T(/ACAIRB/ = Yes) WHEN (!Doppler up! AND !CA stage! completed)

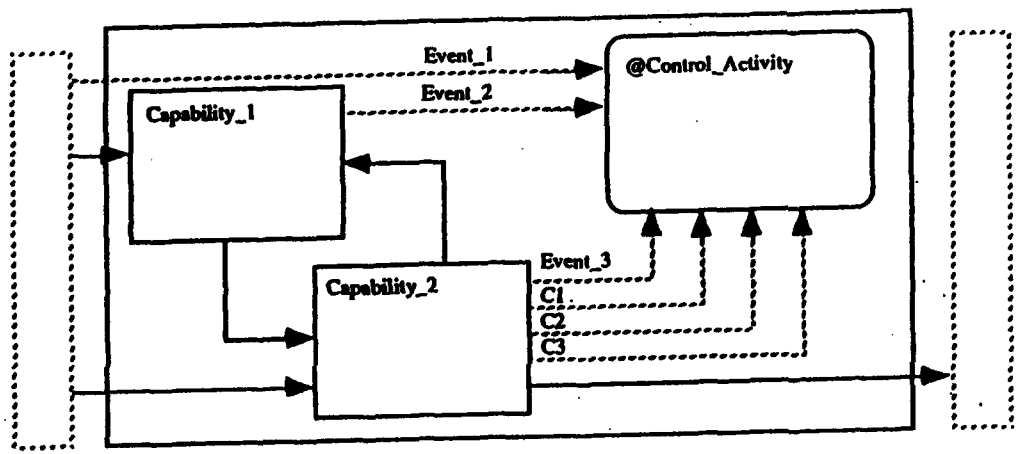
!Doppler up! /DRSREL/ = Yes AND /DRSMEM/ = No AND !Doppler Reasonable!

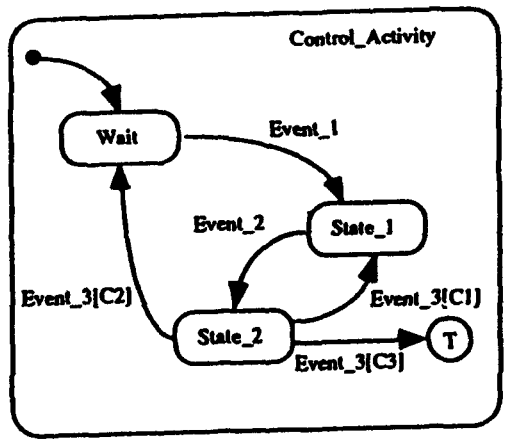
!Doppler reasonable! /DRSREL/ = Yes for last 5 seconds AND
256 fps lseq /DGNDSPD/ lseq 1456 fps AND
/DGNDSPD/ change from .2 sec ago lseq 50 fps AND
/DRFTANG/ lseq 29.5° AND
/DRFTANG/ change from .2 sec ago lseq 4° AND
at least 1 bit change in both /DGNDSPD/ and /DRFTANG/ in last second

MPDS Operating Modes



ECSAM Process Description

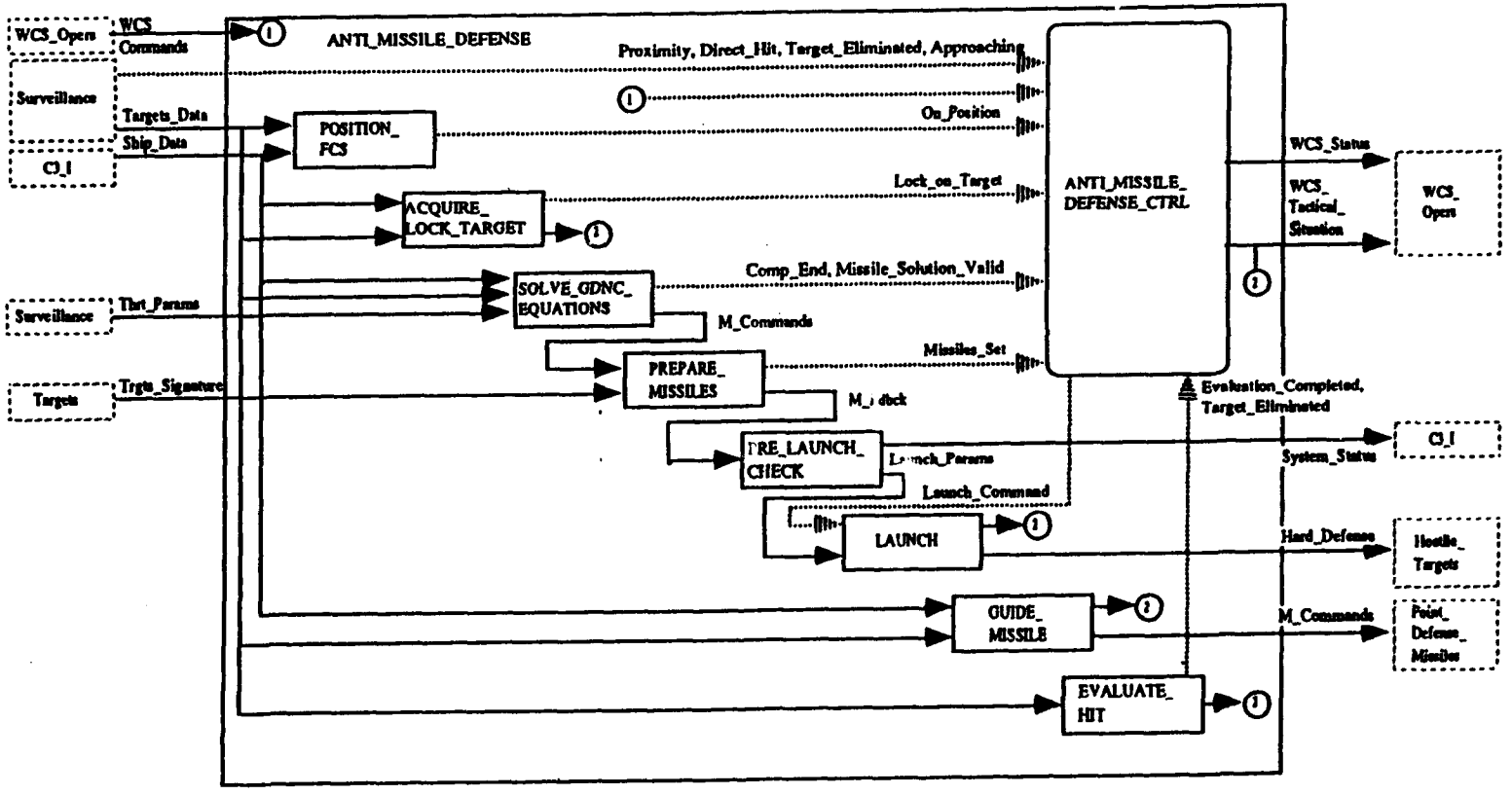




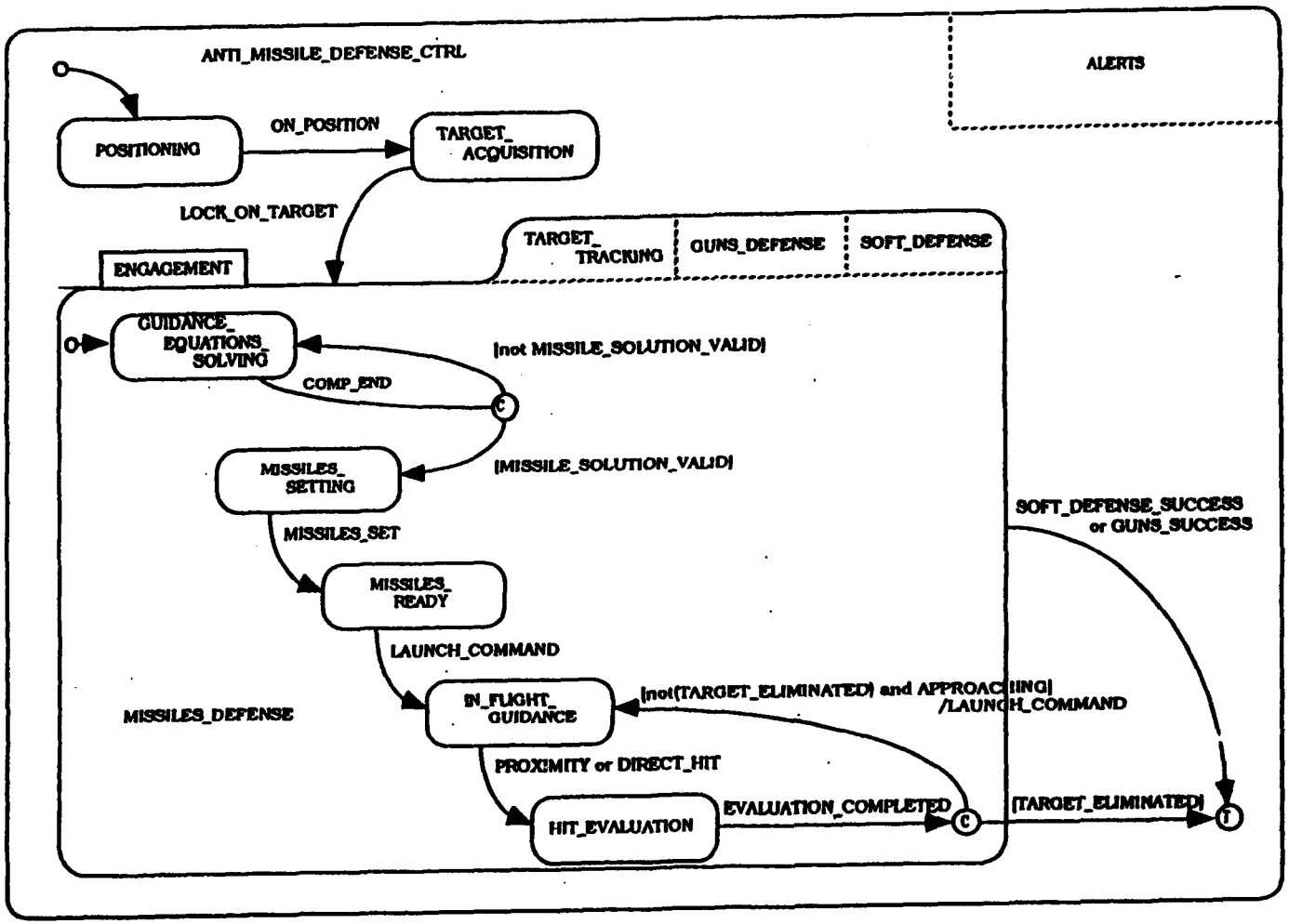
Activities (Capabilities) / States Correlation

State \ Activity	Capability_1	Capability_2
Wait		
State_1	√	
State_2		√

MPDS ANTI MISSILE DEFENSE Process DFD



MPDS ANTI MISSILE DEFENSE Process Control Activity

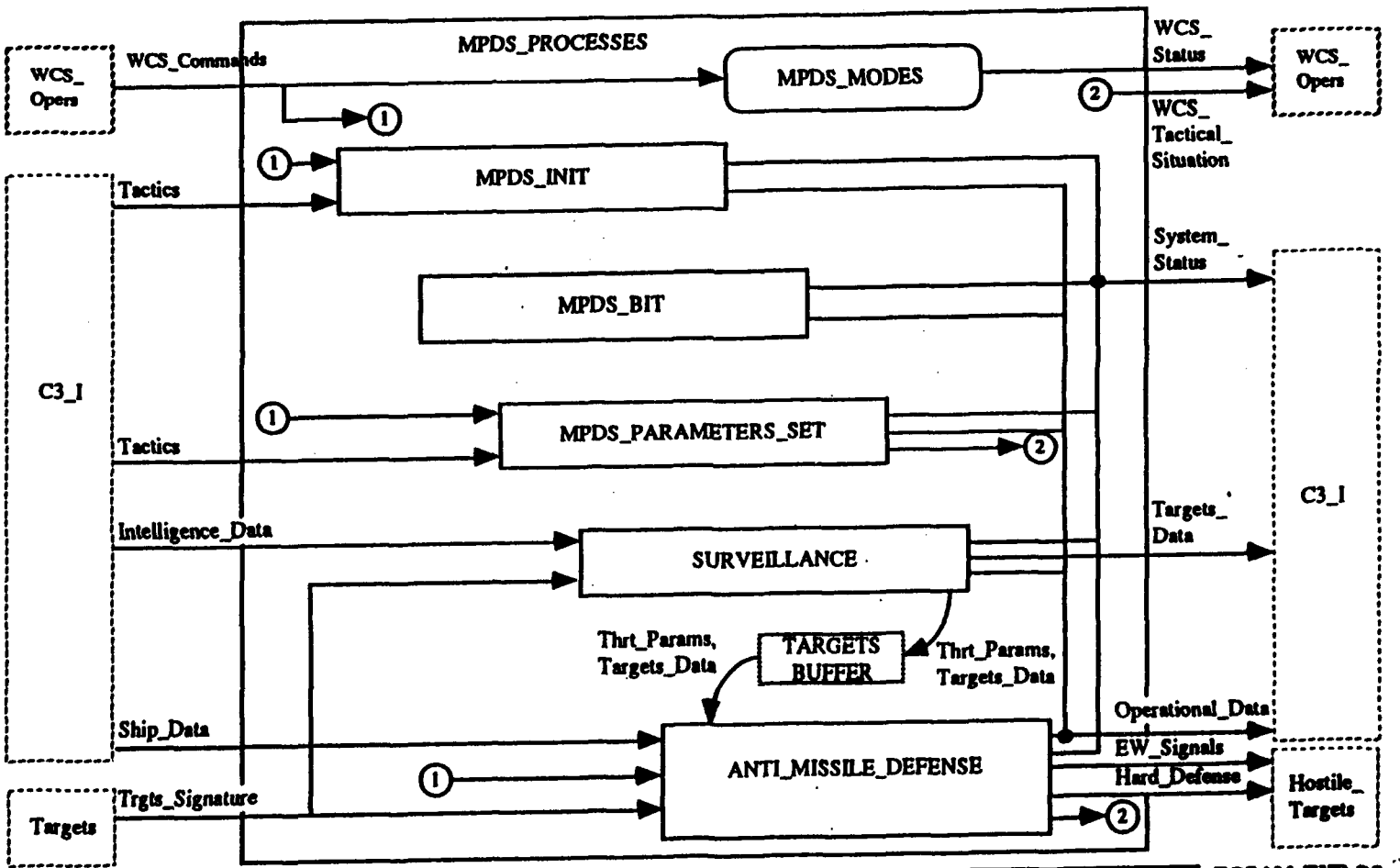


ANTI MISSILE DEFENSE PROCESS
Sample Capabilities / States Mapping

State	Activity	Sub-System
Positioning	Position_FCS	WCS
Target_Acquisition	Acquire_Lock_Target	WCS
Soft_Defense	Deceive_Agressor	EW
	Check_Deception_Success	EW
Guidance_Equations_Solving	Solve_GDNC_Equations	WCS
Missiles_Setting	Prepare_Missiles	PDM
Missiles_Ready	Pre_Launch_Check	WCS
	Pre_Launch_Check	PDM
In_Flight_Guidance	Launch	PDM
	Guide_Missiles	WCS
Hit_Evaluation	Evaluate_Hit	WCS
. . .	. . .	. . .

MPDS Functional Capabilities

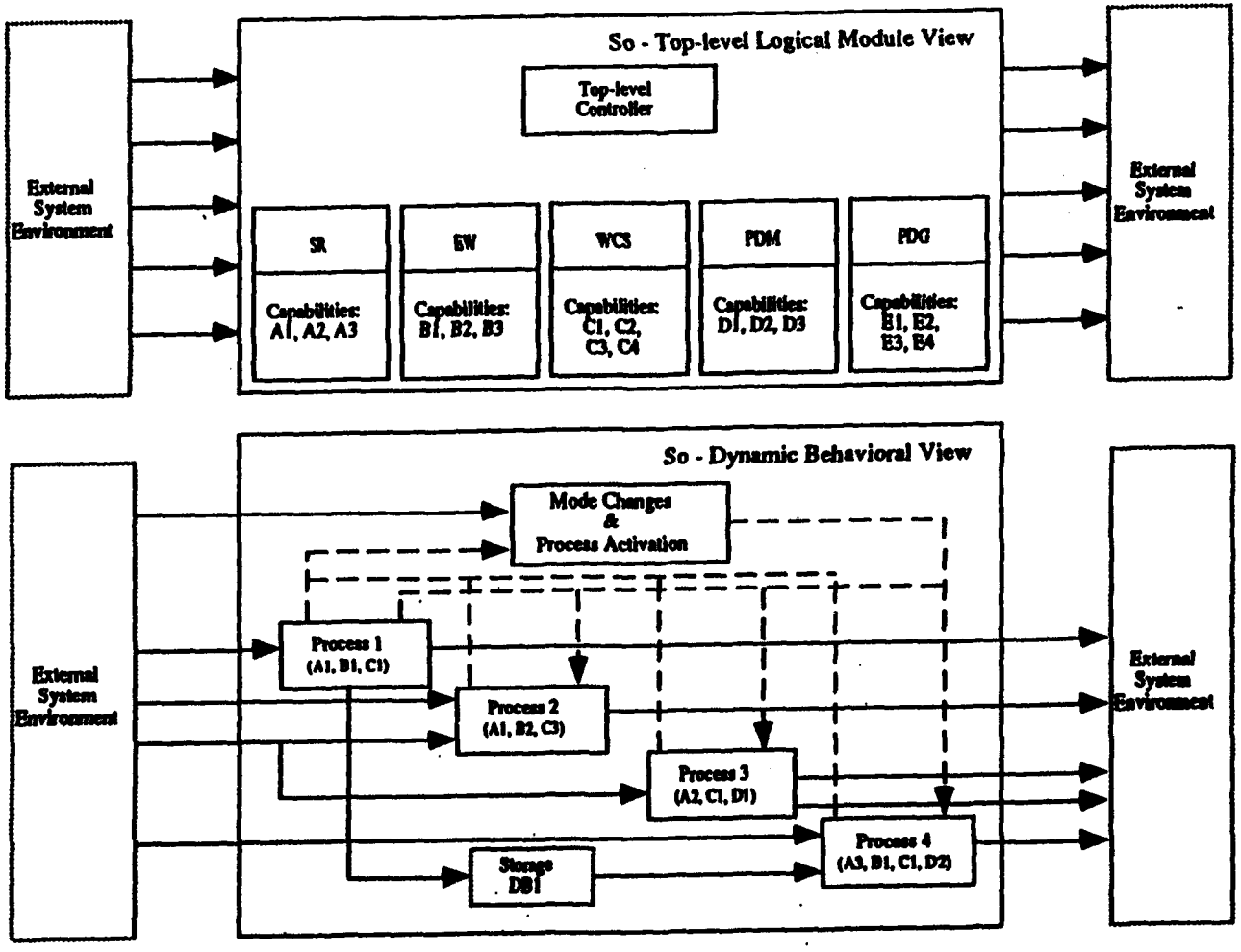
- Functional capabilities are represented by processes



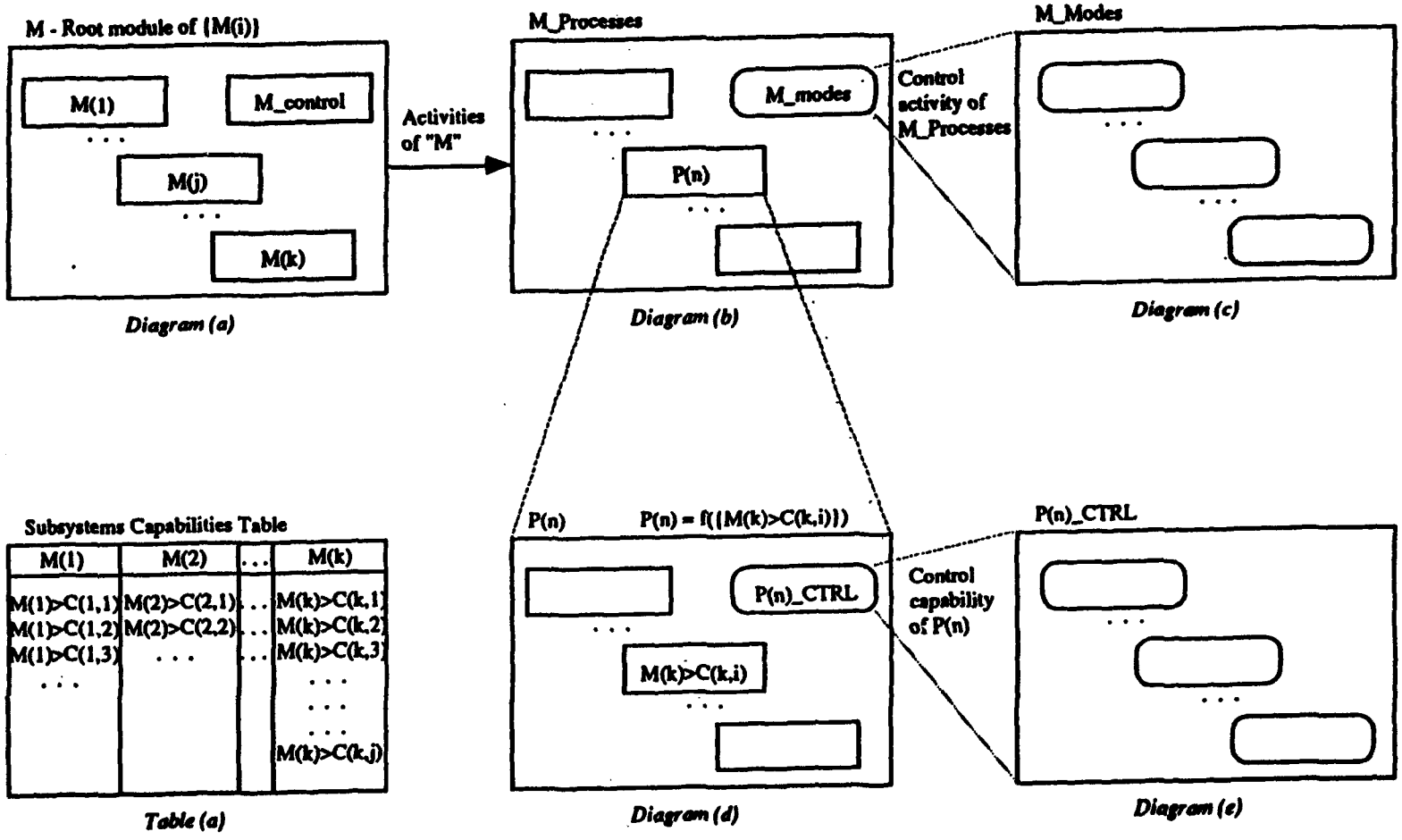
MPDS Processes and the Modes in Which They Are Active

MPDS Modes / MPDS Processes	Stand_ by	Init	Para- meters_ Set	Operatio_ nal
MPDS_Init		√		
MPDS_Bit	√			√
MPDS_Parameters_Set			√	
Surveillance				√
Anti_Missile_Defense				√

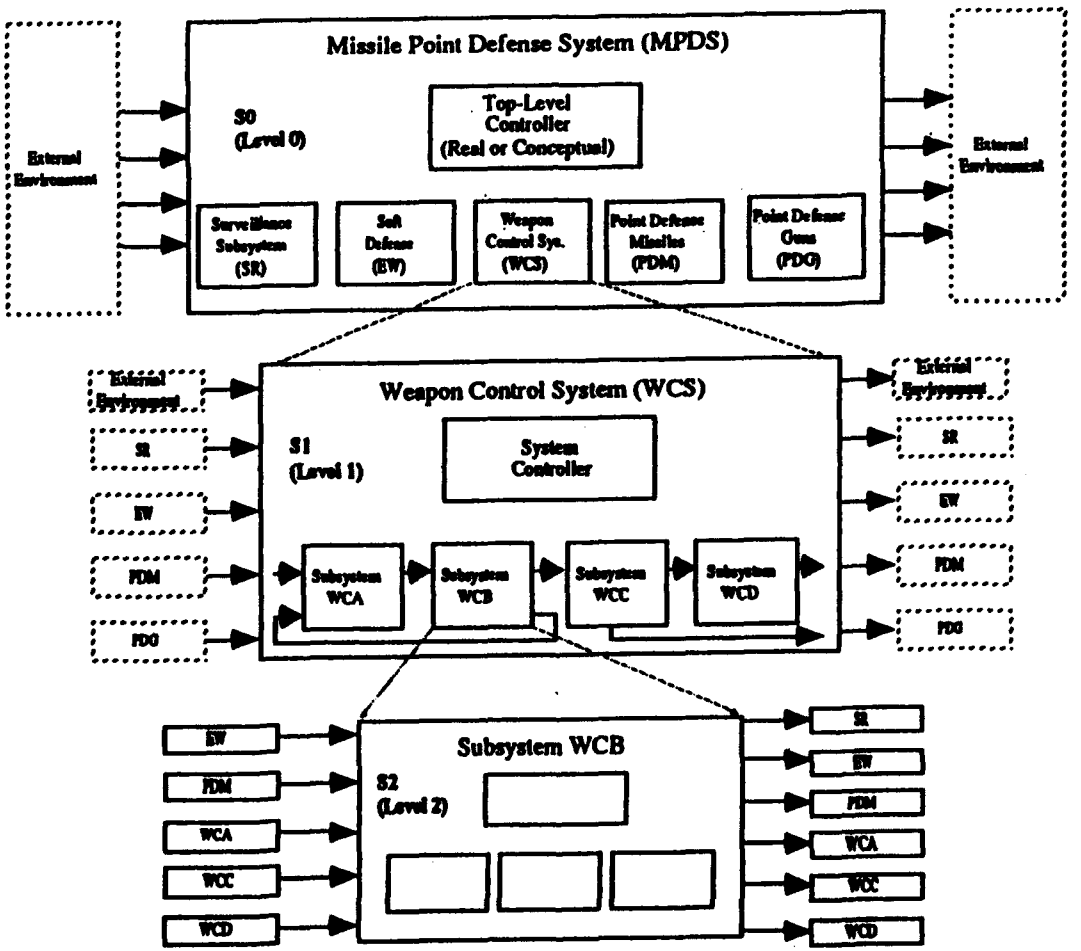
Relationships Between ECSAM Conceptual Views



Relationships Between ECSAM Conceptual Model Components



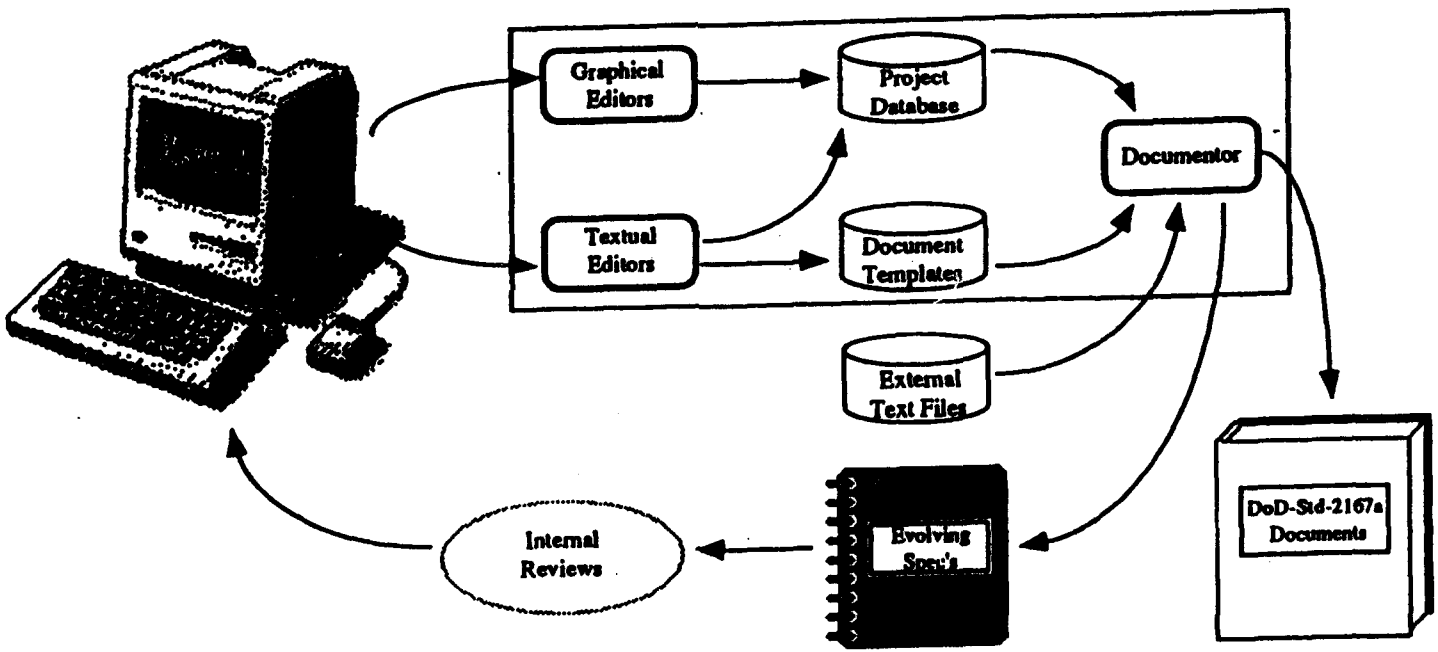
ECSAM Description of Multi-level Systems



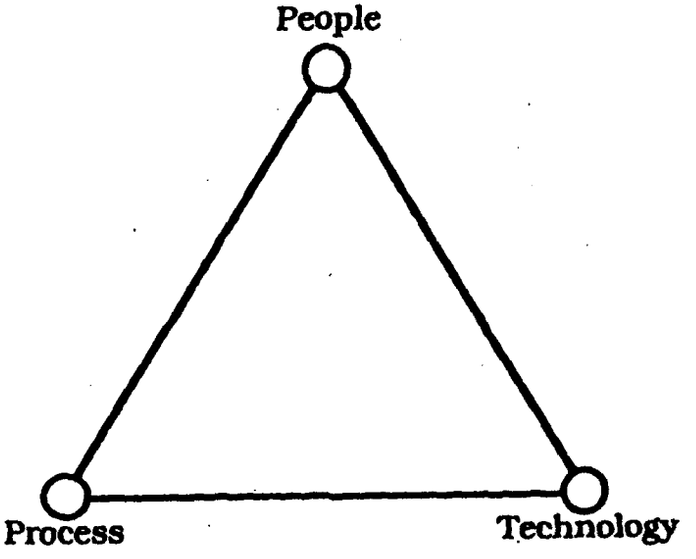
Reuse of Conceptual Subsystem Specifications

- The development of a system's conceptual model specification is one of the most expensive activities in the project's development life-cycle
- ECSAM supports and encourages reuse
- ECSAM conceptual models can be reused to create families of product models
- Conceptual models can be mapped to various system architectures
- Reuse of conceptual models allows rapid specification of new systems within product families

Automatic Document Generation



People, Process, and Technology in Perspective

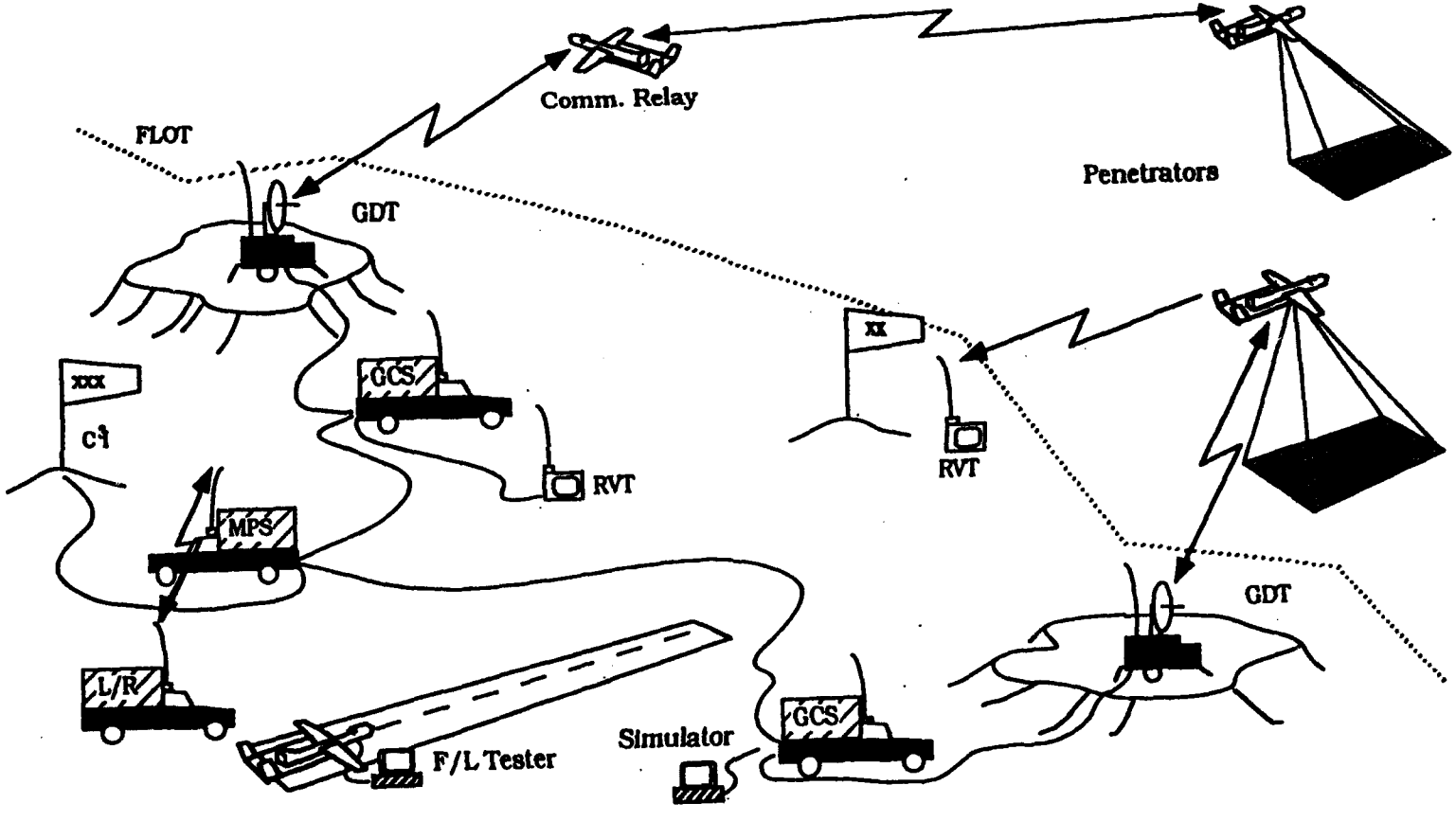


Major determinants of software cost, schedule, and quality performance

Case Histories

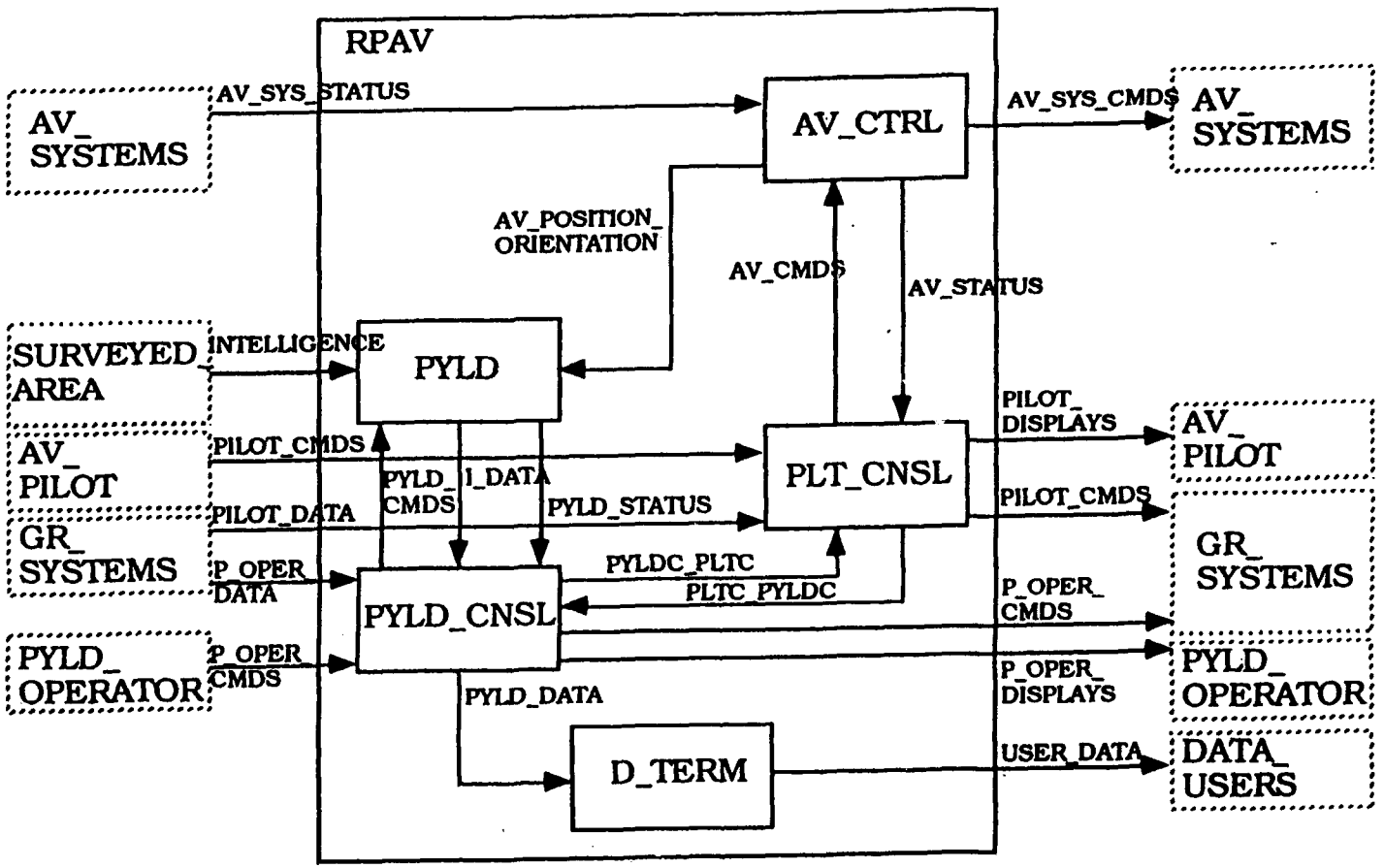
"Hunter" Unmanned Air Vehicle

ECSAM / STATEMATE Application Example

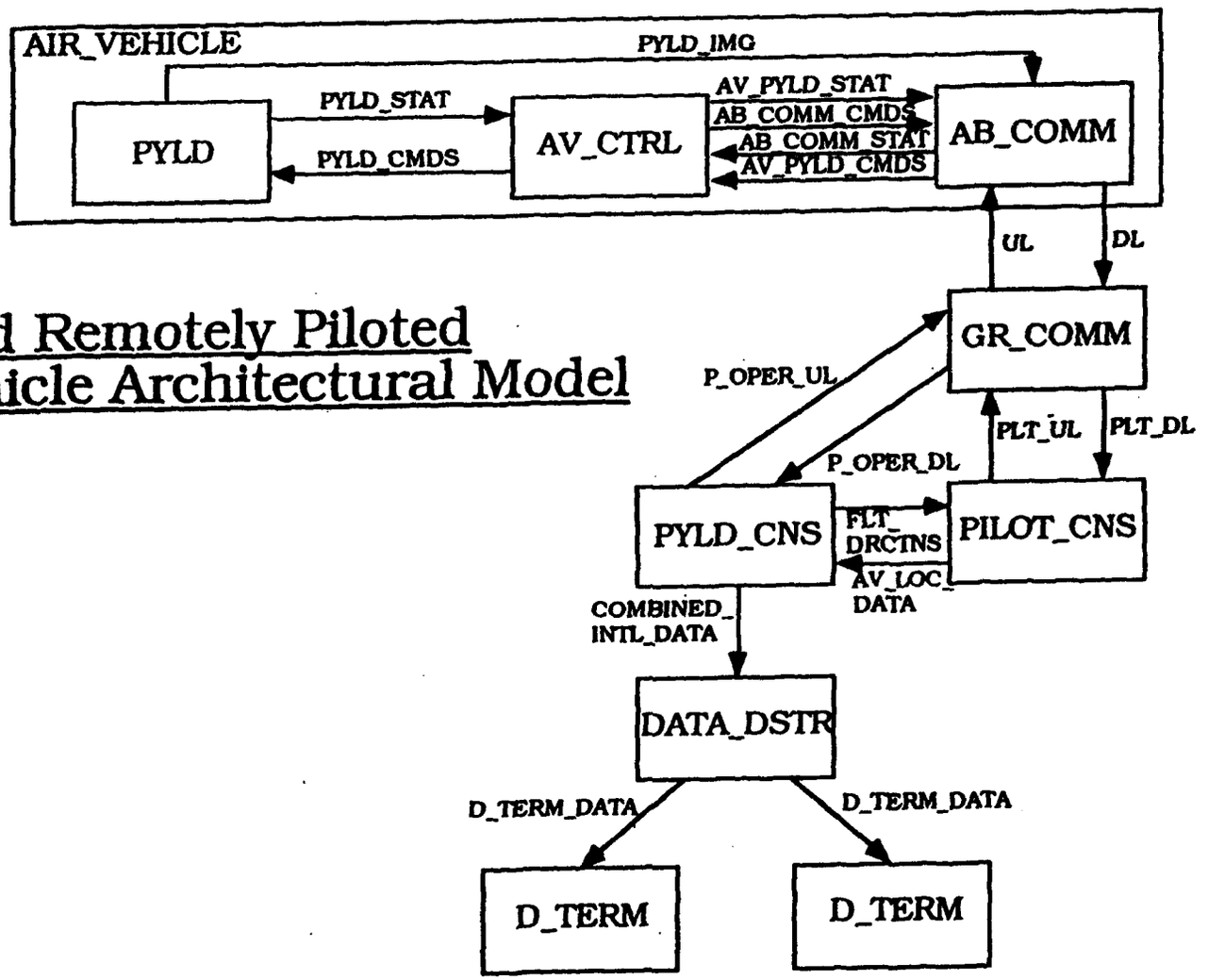


Artist view - not to scale

Simplified Remotely Piloted Air-Vehicle Logical Model



Simplified Remotely Piloted Air-Vehicle Architectural Model



System Description

- A fairly complex system composed of airborne and ground-based subsystems
  - Airborne elements include "Penetrator" and "Relay" unmanned air vehicles
  - Ground elements include mission planning, ground control, and launch/recovery systems, and communication systems
- System characteristics:
  - 8 embedded computer subsystems
  - 134 integrated processors
  - 850K lines of code
  - 215 man-years development effort

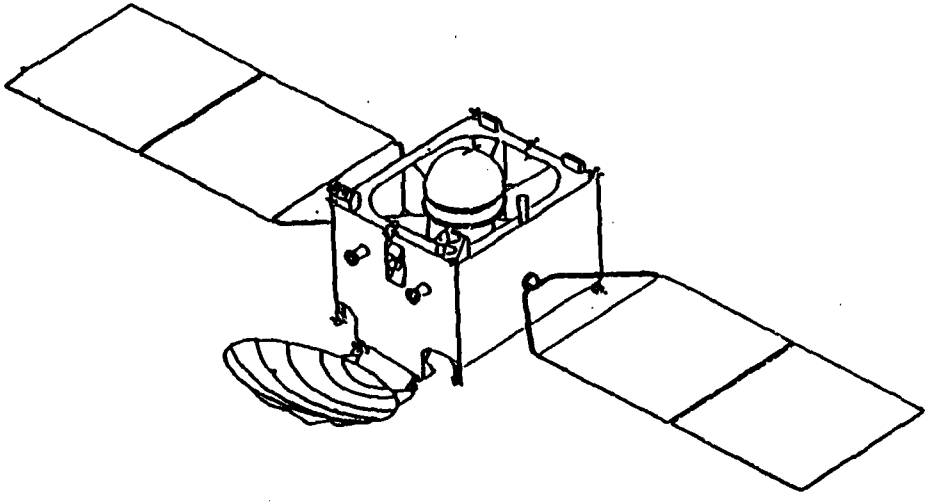
"Hunter" Software Development Methodology

- Evolutionary, structured, rapid prototyping development process
- Development of the Computer Based System (CBS) model using ECSAM
- STATEMATE tool used to define the CBS model and to generate documentation
- Automatic generation of software documentation subset:
  - » Top Level Specifications (TLS) of subsystems
  - » Software Requirements Specifications (SRS)
  - » Interface Requirements Specifications (IRS)
  - » Interface Design Documents (IDD)

"Hunter" Software Phase I Summary

- 500 allocated software requirements
- TLS documents for 15 subsystems
- 8 SRS and 8 IRS documents delivered
- 4 main iterations of software specifications
- Database contains over 5000 data items
- 2500 pages of DoD-STD-2167A documentation automatically generated
- Successfully passed formal IV&V and FCA reviews

"Amos" Geo-synchronous Communication Satellite



Characteristics

- Complex logic of on-board computer software
- Due to economic constraints, reliability and availability are extremely important
- Required life span in orbit > 10 years
- Constraints:
  - » Severe correctness requirements (especially in ROM SW)
  - » Rigid development schedule
  - » Computer memory constraints
- SRS to serve as basis for test and evaluation
- Need for a method and a tool that would enable the systems engineering group to develop system and SW requirements
- Importance to characterize the dynamics of processes controlled by software

"Amos" Geo-synchronous Communication Satellite Development Process

- "Intuitive" development of software requirements specification by the software development team
- Development of an ECSAM model of the embedded spaceborne computer and its software by the systems engineering and software teams
- ROM software requirements developed
- System requirements allocated to software model
- Static analysis of the ROM software ECSAM model completed
- Dynamic simulations of the ROM software to begin shortly
- Specification and analysis of the RAM software to begin shortly, based on the ROM software model

The "Revelation"

"Everyone knew 'exactly' what had to be done, until someone wrote it down ..."

[R. Pressman]

Scope of Work

- Over 100 system requirements allocated to SW
- 17 dynamic processes
- 7 main logical subsystems (total of 11 logical subsystems)
- Automatically generated ROM SW SRS is 500 pages long

Conclusions

- The use of ECSAM and STATEMATE with massive in-house methodological support served as a strong catalyst in the creation of requirements specifications
- The ECSAM method helped to create a common foundation for the specification of software requirements by the systems and software engineering groups
- Top-Level Specifications (TLS) documents are useful to the developers of the requirements model (conceptual model)
- To be effective, users have to undergo thorough training in the use of the method (ECSAM), tool (STATEMATE), and database containing the model
- Maintenance and update of documents can be done only by qualified personnel - "wasting" engineers' time

Avionics Upgrade Project

Project Objective

To upgrade the avionics of aging aircraft, utilizing existing software wherever possible

Approach

- Formulate system requirements allocated to software, and derived software requirements
- Build conceptual system and software models
- Allocate requirements to modules
- Manually "inject" code from existing avionics systems into new software modules wherever feasible
- Extract design and interface definitions from existing code and "inject" them into new design documents
- Generate up-to-date documentation

STATEMATE/ECSAM Use Background

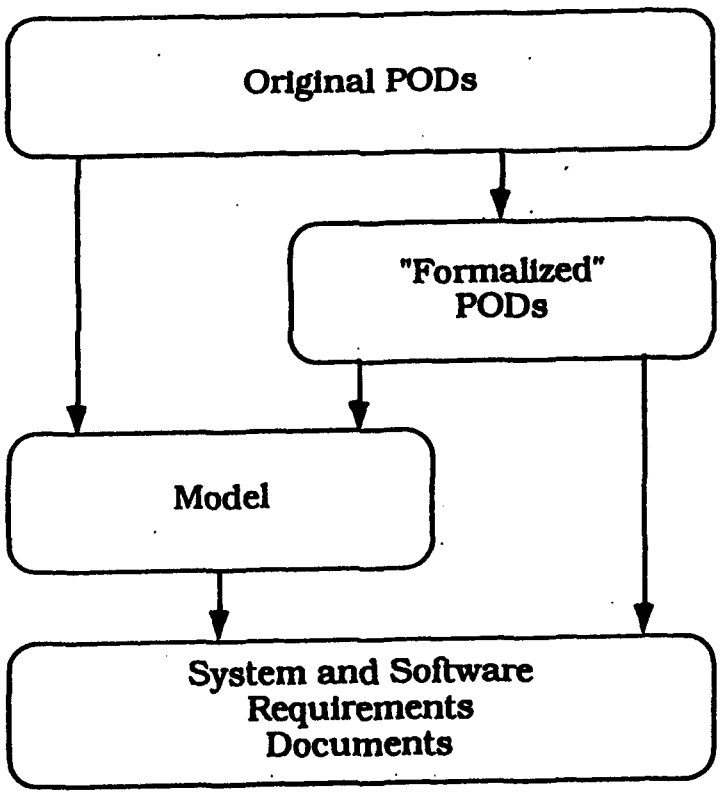
- "Lahav" plant is developing "families" of avionics systems
- Description of system dynamics by statecharts is a part of typical development processes
- Requirements for the institutionalization of the method and tool exist. The tool and method are required to support:
  - » Easy adaptation to various projects and rapid response to RFPs
  - » Re-use of existing software in the construction of new systems (reverse engineering)

Existing Documentation of Previous Generations

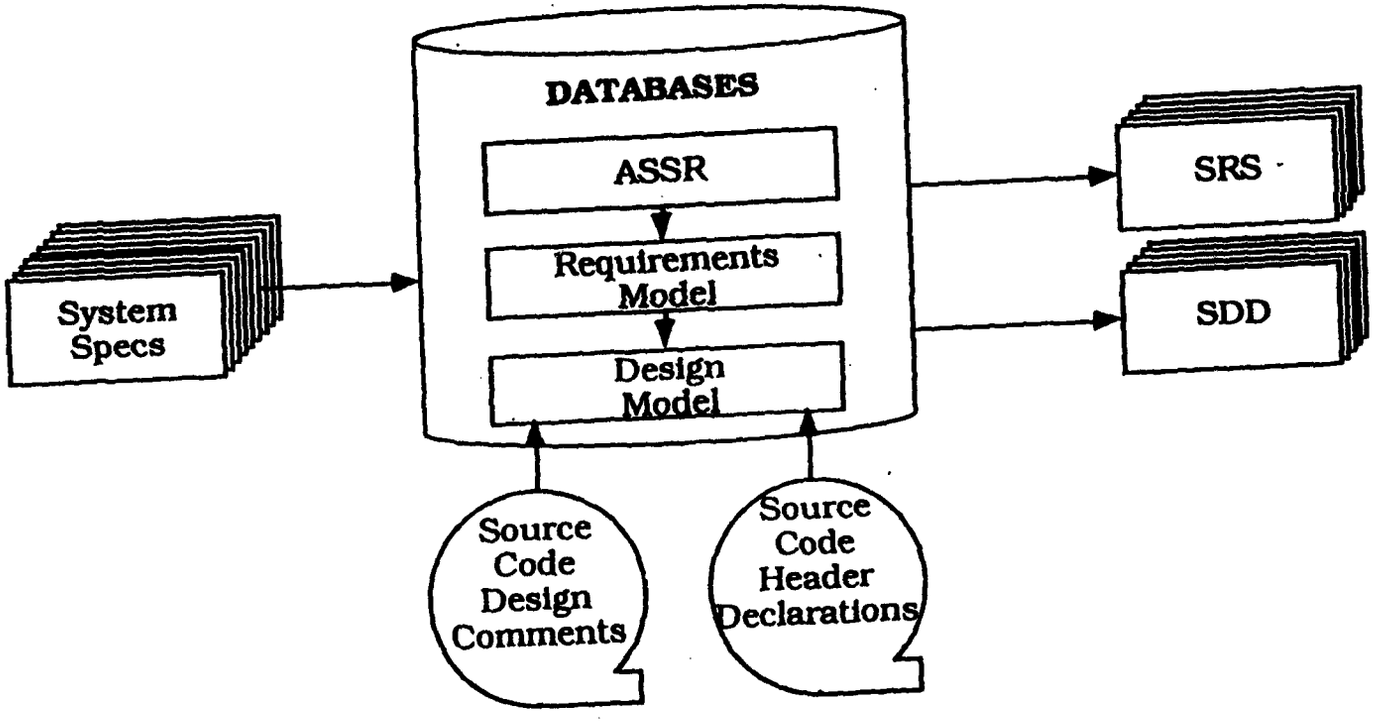
The following existing documentation is available:

- Pilots' Operational Definitions (PODs)
- External system processes described by statecharts prepared on a word-processor ("formalized" PODs)
- Design comments embedded in old code
- Inter-module interfaces definition extracted from old code (checked by compilation)

Conceptual Model Development



Development Process



Avionics Development Highlights

- Development based on Pilot Operational Definition (POD) documents
- Conceptual and design models developed
- Low-level design described by means of code lifted from existing software, embedded in the ECSAM design model

Conclusions

- Use of ECSAM and STATEMATE enabled:
  - » optimal reuse of existing SW
  - » computer-aided organization of resources and products
- Use of system and SW level reviews provided for early detection and correction of defects
- Development based on the lines described above necessitates planning of resource allocation (time and personnel) during early project phases

Summary

- ECSAM is an Object-Based, model-driven development approach
- Use of three well-defined views separates concerns, simplifies anlysis and assures model completeness and execution
- Analysis follows a well-defined process
- Starting the analysis with a top-down engineering process, rather than a bottom-up modeling approach, simplifies and expedits development
- Use of conceptual models allows reuse
- Many architectural implementations can be developed for various mixes of configurations of conceptual models
- Approach taught and applied in many projects

Embedded Computer Systems Analysis and Modeling

The ECSAM Approach

Michael Winokur, Reuven Gallant
Israel Aircraft Industries, Dept. 9100
Ben-Gurion International Airport, Israel 70 100

Jonah Z. Lavi
CBSE Associates
e-mail: lavi@math.tau.ac.il
3 Tabenkin St., Ramat-Gan, Israel 52 302

Joseph Kudish
Systems and Software Engineering
e-mail (on DIALCOM): 152:CNM428
P.O. Box 492, Rehovot, Israel 76 106

Abstract†

ECSAM is a comprehensive object-based model driven method for the analysis and modeling of Computer Based Systems (CBS) and their software. It was developed at the Israel Aircraft Industries (IAI) for the analysis and design of complex embedded computer systems and their software. Using this method systems are described by two complementary models: a conceptual model and a design model. The conceptual model describes a system in terms of its conceptual structure and interfaces, its capabilities and its dynamic behavior. The design model describes the systems in terms of its architecture, hardware and software subsystems, communication channels and their properties, the human - machine interface design and implementation constraints.

ECSAM relies on graphical representations of all its views. It uses formal semantics wherever applicable thus allowing the testing of the systems' static specification and the simulation of its dynamic behavior.

ECSAM is used for the static and dynamic analysis of complex embedded computer systems. It is applicable to simple, single computer systems as well as to complex multi-systems and their software. It has been developed since existing methods have not addressed satisfactorily the development needs of modern CBS at IAI.

The method has been developed since 1980 answering the needs of IAI projects. The evolving versions of ECSAM have been taught over the years to several hundred system and software engineers. Currently the method is being successfully used by many projects. The method is supported by the STATEMATE CASE tool.

The paper describes the ECSAM conceptual model, outlines the characteristics of the design model and the mapping between the two models, and briefly outlines the IAI experience in the method's use.

† The paper was submitted to the ICSE 16 conference

Index Terms: *Analysis, Computer Based Systems Engineering, Conceptual, Design, Dynamic, ECSAM, Embedded Computer Systems, Event Driven Systems, Executable specifications, External, Internal, Models, Multi-Systems, Reactive Systems, Requirements, Software, Specification, Statecharts, STATEMATE, Systems.*

1. Introduction

Modern Computer Based Systems (CBS), particularly embedded computer systems, are often very complex. Many of them are hierarchical multilevel systems that contain many computers in each level. Others are composed of many loosely coupled cooperating subsystems. Frequently, each of these subsystems is large and complex by itself, and is implemented as a multilevel multicomputer system. As a typical example we may take a modern integrated avionics system, which is composed of many subsystems, e.g. navigation, radar and electronic warfare systems. Each of these subsystems typically is a multicomputer system. Most of them react dynamically to random external signals and sequences of external events. Their dynamic behavior also depends on the operational history of the systems and on complex logic conditions.

Specification languages and analysis methods applicable to such systems should allow the development of coherent, precise models of the systems being analyzed and specified. They should consider the static characteristics and the dynamic behavior of the systems, their multisystem and multilevel characteristics and the relevant design issues. The methods should use "formal" graphical representations of the system models, based on well-defined semantics, to achieve clarity, precision, and testability. The methods should allow a stepwise iterative analysis, design and implementation of the systems and their software and should be teachable to a wide spectrum of systems and software engineers.

Many methods for the description, analysis and specification of computer based systems and their software are known in the literature [e.g. ALFO85, ASCE89, HATL87, HENI80, ROSS85, RUMB91, WARD86, ZAVE84]. Previously, the majority of the methods supported only the static analysis of systems, such as functional decomposition or data flow techniques. Others related to problems associated with the dynamic behavior of systems and software. Very few relate to the analysis and specification of entire computer based systems and their software. Only few of the object-oriented software analysis and design methods published in recent years address the static and dynamic aspects of software [BOOC91, RUMB91]. Methods addressing software, however, usually disregard system aspects, and treat software as if it existed in "vacuum".

This paper describes ECSAM, an Embedded Computer Systems and Software Analysis and Modeling method. Using this method a model of the system to be developed is built following an orderly process obeying a well defined set of rules. The essential idea behind ECSAM is that the driving force behind the development process is the construction of coherent models that represent the problem and solution spaces of the desired system. The model of the problem space is called a "conceptual model", while the solution space model is called a "design model".

ECSAM, an Object-Based method [BOOC91], supports the analysis of entire Computer Based Systems as well as the analysis of software systems and modules. It addresses the issues of multi-level systems and multi-systems, a capability not adequately addressed in other methods [DAVI90]. ECSAM is especially suitable for the analysis of embedded systems such as process control systems, avionics systems, military systems, medical instruments and modern consumer electronic systems. It is particularly well-suited for software design intended for implementation in Ada language, which has been successfully demonstrated in several projects. Graphic techniques are used to produce the models and to represent their views. ECSAM provides guidelines for the exploitation of graphical representation that provide a simplified initial view of a complex system. As the system concepts mature and stabilize, the representation evolves in a way that promotes incremental increase of the depth of understanding of the system specification.

ECSAM is being developed, by the authors at the Israel Aircraft Industries (IAI) since 1980 [LAVI84, LAVI86, LAV881, LAV882, LAVI89, LAVI91, LAV921, KUDI92, WINO90, WINO91]. Initially the efforts were concentrated on analysis of the software of embedded computer systems. Later on, while working on multicomputer systems, the scope was extended to address the engineering of systems as well as their software. Recently it was further extended to address the development of multisystems and to the analysis of the systems' external dynamic behavior. The method is being augmented and improved continuously based on practical experience gained during its application to projects within IAI. The ECSAM development was the basis for the development of STATEMATE [HARE90], a CAS^{2}E [1] tool used in conjunction with the method at IAI. ECSAM has proven useful to engineers in organizing their thoughts and expressing them in an explicit and clear way thus improving considerably the analysis process and the quality of the specifications of computer based systems. ECSAM has been taught during recent years to several hundred system and software engineers. The method and the supporting tool STATEMATE are being used successfully in many projects within IAI.

This paper consolidates ideas published in previous ECSAM papers and new unpublished work done during the past three years. It focuses on the ECSAM modeling approach. It describes the ECSAM conceptual and design models and their views, the generic model of embedded computer systems, the development of multilevel and multisystem models and basic concepts of the mapping between the conceptual and the design models. It summarizes the experience of applying ECSAM in projects at IAI and concludes with a short discussion comparing the ECSAM model with other known CBS analysis models.

2. The ECSAM Modeling Approach

The development of engineering artifacts typically requires two augmenting complementary models: a conceptual model and a design model [JACK92, LAVI93]. The conceptual model which describes the system in the problem space is necessary for the understanding of the physical phenomena on which the system is based. Its main purpose is to support the behavioral and performance analyses of the systems. The design model describes the system in the solution space. It represents the actual structure of the system and supports its design process.

The ECSAM modeling approach addresses the conceptual and the design models and defines the mapping between them. As will be discussed later, the conceptual model can describe a family (class) of systems which can be mapped to different architectures with similar functionality depending on implementation constraints. Both models are used concurrently to analyze and engineer the System Under Development (SUD).

According to Figure 1, ECSAM can be viewed as a prism, resolving the system specification into two models - the design and the conceptual ones, further resolving the conceptual model into interrelated views.

Due to the complexity of embedded computer systems it is impossible to describe their conceptual models by a single view. Instead, it is necessary to separate concerns during the system analysis process and utilize different

---

1 Computer Aided Systems and Software Engineering

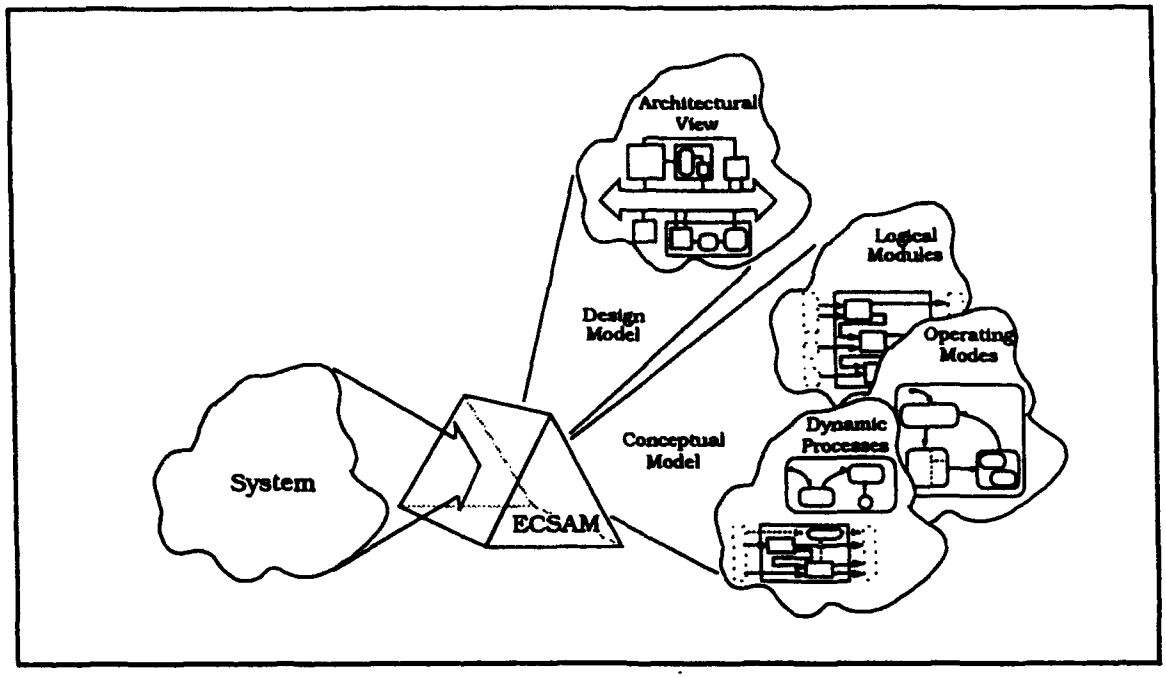


Figure 1 - ECSAM Model components

related modeling views to describe the static and dynamic aspects of the system, as shown in Figure 1.

3. The ECSAM Conceptual Model

The ECSAM conceptual model is described by the following three views:

- The **Logical Modules View**, which describes:
  - The partitioning of the system into its logical subsystems
  - The external information that flows between the system and its environment
  - The information that flows between the internal subsystems
  - The functional capabilities (activities) performed by each logical subsystem.
- The **Operating Modes View** which describes the system's main operating modes and the transitions between them.
- The **Dynamic Processes View** which describes the behavioral processes that occur in the system in its various operating modes in response to external or internal events.

The overall dynamic behavior of the systems is jointly described by the last two views.

The three views are complementary and interrelated, just like views used in the drafting of mechanical artifacts. During the systems analysis we iteratively examine the different views assuring mutual consistency and completeness of the model and of the resulting specifications.

The ECSAM views and concepts discussed in the following sections are demonstrated using a simplified naval anti-missile point defense system (MPDS) as an example. The MPDS is used for the detection of attacking missiles and for defense against them.

This chapter first presents each of the conceptual model views and then the relationships between the views.

The Logical Modules View

This view describes the partitioning of the system into logical modules or subsystems, information flows, and functional capabilities performed by each of the subsystems. The logical modules can be viewed as abstract (or virtual) machines used as building blocks in the modeling process. Logical modules are implementation independent. Their capabilities can be allocated to hardware, software, manual operations or a mixture of all three. As will be discussed later, their capabilities may even be allocated to various architectural modules due to implementation constraints. This definition follows the information hiding concepts described by Parnas [PARN86]. At the pure software level, ECSAM's logical modules are abstract data-types that have internal states and data structures. The common interpretation (or rather implementation) of logical modules in Ada are packages.

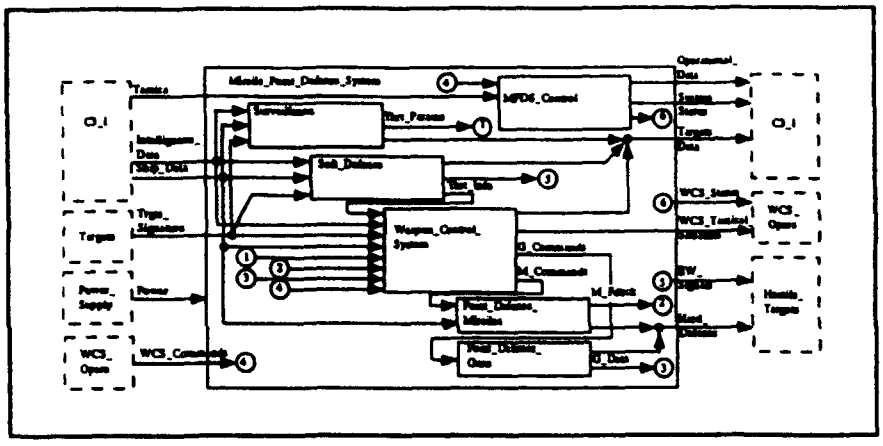


Figure 2 - MPDS Logical Modules

A diagram of the MPDS logical modules view is shown in Figure 2. In this diagram, the system, its external (environmental) and internal logical subsystems (modules) are described as rectangular boxes. The labels on the lines connecting the boxes represent the main information flows between them. Each information flow may contain several levels of lower-level flows and information items. The constituent information flows are not presented graphically for the sake of clarity, but nevertheless, they are an integral part of the model. Lower level information flows can be observed, for example, by making queries to the supporting tool database. Information flows may represent process data flows or control flows. Information flows connect the source and sink of the information, disregarding its physical routing through the system. It must be stressed that the lines connecting the logical modules do not represent hardware signals or digital communication paths in the system, that are shown in the system's architectural design diagrams.

The MPDS environmental systems shown in Figure 2 are systems (external to the MPDS) with which the MPDS system interacts. These systems are:

- C^3I - An external command and control system
- Targets - Unidentified targets which can be classified as friendly or hostile or can remain unidentified
- Power_supply - An external system which supplies electrical and hydraulic power to the MPDS
- Hostile_Targets - Targets classified by the system as hostile

Environmental systems may send signals (messages) to the system, receive signals from it, or do both. In the latter case the environmental systems are drawn twice - to the left of the system (as sources), and to its right (as sinks). This simplifies the drawings, and their comprehension and analysis.

The MPDS can be decomposed into the following logical subsystems:

- Surveillance, e.g. a radar or an electro-optical system
- Soft_Defense, e.g. an electronic warfare system
- Weapon_Control
- Point_Defense_Missiles
- Point_Defense_Guns

The function of the MPDS_Control subsystem (the "System Controller") depicted in the diagram will be explained in the section dealing with the relationships between the conceptual views. Some of the internal information flows are also shown in Figure 2. The internal information flows, "hidden" inside the model provide an important input to the analysis and validation of the model's completeness and correctness.

The functional capabilities performed by the system are shown in Figure 3. Some of the functional

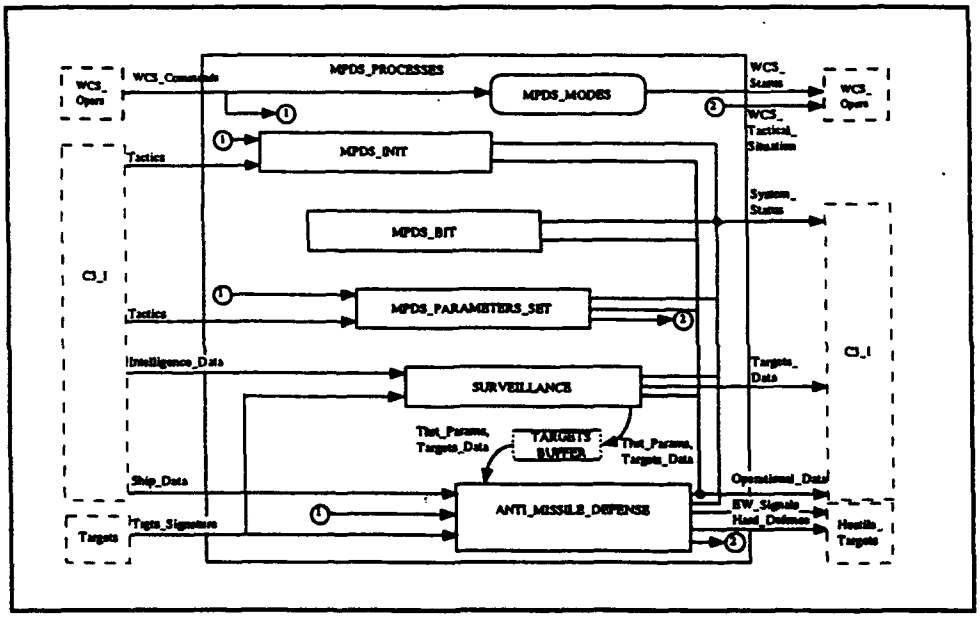


Figure 3 - MPDS Functional Capabilities

Surveillance Radar (SR)	Soft Defense (EW)	Weapon Control System (WCS)	Point Defense Missiles (PDM)	Point Defense Guns (PDG)
Detect_target	Deceive_agressor	Position_FCS	Prepare_missiles	Prepare_guns
Identify_target	Check_deception_	Acquire_lock_target	Pre-launch_check	Control_aim
Create_target_track	success	Track_target	Launck	Fire_guns
Update_target_track	. . .	Solve_gdnc_equations	. . .	. . .
. . .		Pre_launch_check		
		Evaluate_hit		
		Guide_missile		
		. . .		

Table 1 - Sample Capabilities of Anti Missile Point Defense Subsystems

capabilities of the subsystems are listed in Table 1. The relationship between the capabilities of the system and the capabilities of its subsystems are described by the dynamic processes described later on. Since the logical modules are active objects [BOOC91], the functional capability charts always include a control block (MPDS_MODES in this diagram). Its function is to control the order in which the capabilities are activated, as will be described later on.

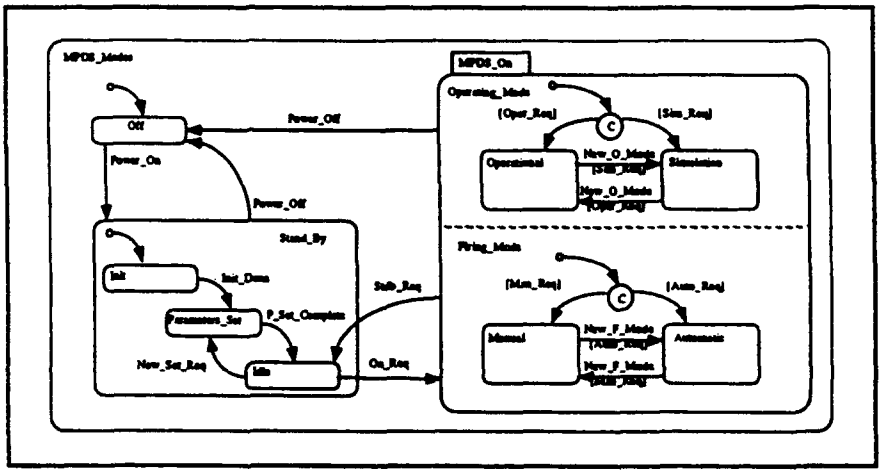


Figure 4 - MPDS Operating Modes

Some of the functional capabilities of the MPDS are:

- MPDS_INIT - Initialization
- MPDS_BIT - Built-in test
- MPDS_PARAMETER_SET - Parameters setting
- SURVEILLANCE - Surface and air surveillance
- ANTI_MISSILE_DEFENSE - Defense against attacking missiles

The two diagram types represent different view points on the system. The functional capabilities chart shows the activities performed by the system, i.e., "what" the system is doing, while the logical module chart describes the conceptual structure (the conceptual "how") of the system in terms of logical subsystems or building blocks. The use of logical subsystems in ECSAM, gives it its power in the rapid development of system models and the reuse of conceptual models of various subsystems.

The Operating Modes View

Modes are externally visible generalized system states, used to simplify the system description and analysis, in the sense described by [HENI80]. A mode-by-mode description of the system is a simple way to characterize top level states of systems as seen by their operators or external observers. The operating modes concept has been used very effectively by engineers and operators of systems for many years in various fields of engineering. Their definition in the initial phases of analysis is very important since they characterize one of the system's major operating features.

Modes and the transitions between them can be described by state transition diagrams. In their conventional flat form, such diagrams typically are not structured, do not support the concept of concurrency, are difficult to read, and are not easily comprehensible by the

system's customers. ECSAM alleviates these problems by adopting the Statecharts approach, developed by Harel while consulting for IAI [HARE87]. Statecharts employ the notions of multi-level structure (state embedding), levels of detail, and the ability to split states into concurrent "and" components. They allow the specification of concurrency, independence and synchronization in various ways and at all levels. The advantages of using Statecharts compared with conventional state transition diagrams are discussed by Harel in his paper [HARE87]. The use of Statecharts has become a common practice, and their syntax and semantics won't be elaborated in this paper.

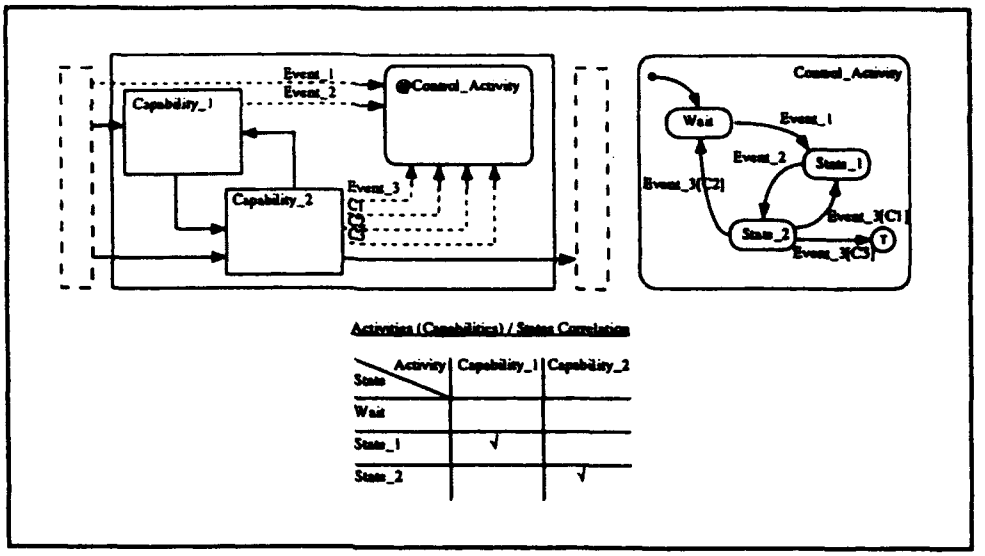


Figure 5 - ECSAM Process Description

The main operating modes of the MPDS are shown by the Statechart in Figure 4. According to this figure, the system can be in the "Off" mode, "Stand_By" mode or the "MPDS_On" mode. When in the "MPDS_On" mode, the system can be in the "Operational" or "Simulation" mode and simultaneously be in the "Manual" or "Automatic" Mode.

The Dynamic Process View

The dynamic process view describes the sequence of operations, within each functional capability, which are activated in response to external or internal events. The dynamic processes describe two aspects of the systems behavior: the transformation of system's inputs to its outputs by the subsystem capabilities and the logic controlling the order and timing of the activation of these capabilities.

Each dynamic process is defined graphically by two complementary diagrams:

1. The process DFD (Data Flow Diagram), describing the functional capabilities participating in the process and the data flows between them. It also contains a control capability and control signals.

2. The process control diagram, describing the control capability which determines the order and timing in which the process function_l capabilities are activated. It is represented by a Statechart.

Please note that Statecharts are used in ECSAM for two distinct purposes: for the description of the operating modes and the transitions between them, and for the description of control of the dynamics of the system processes.

The combined use of the process DFD and the process control diagram to describe a process is demonstrated in the simple example presented in Figure 5.

Two functional capabilities (Capability_1 and Capability_2), are participating in the process in the DFD on the left-hand side of Figure 5. The process control is described by the Statechart on the right-hand side of the same Figure. The participating functional capabilities are performed by some of the logical subsystems and the control of the process is performed by the system controller.

The process begins in the Wait state. Following the occurrence of Event_1 the process enters State_1. While in this state Capability_1 is active. The correlation between the states and the functional capabilities is represented in a tabular form (and not graphically) as shown in Figure 5 and later on in the example described in Table 2. Upon the occurrence of the internally generated Event_2, the process enters State_2. While in this state Capability_2 is active. Following the occurrence of Event_3, depending on the status of the conditions C1, C2 and C3, the process reenters State_1 or the Wait state, or is terminated as marked by the Encircled "T" symbol.

Analyzing the control Statechart, it is obvious that it describes various threads of dynamic behavior, depending on the order of occurrence of events and the associated conditions. The description presented in the control Statechart is compact in the sense that paths which can be traversed many times (repetitive parts or parts which belong to several threads of dynamic behavior) are "folded" upon themselves, and are presented only once.

To illustrate the power of the dynamic process description we briefly describe a simplified version of the ANTI_MISSILE_DEFENSE processes of the MPDS. This process describes the dynamic behavior of the MPDS

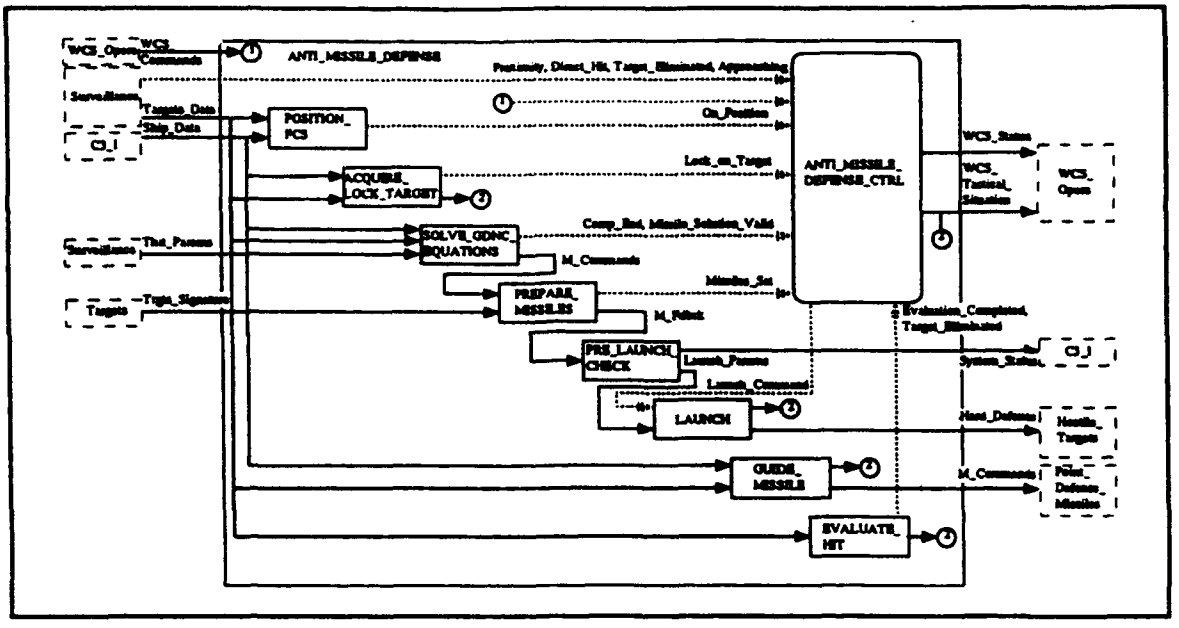


Figure 6 - MPDS ANTI_MISSILE_DEFENSE

functional capability Anti_Missile_Defense specified in Figure 3.

Figure 6 describes a simplified DFD of the process including the control signals. The functional capabilities participating in the execution of this process are a subset of the functional capabilities of the MPDS subsystems specified in Table 1. The diagram is a conventional DFD and will not be further discussed in the paper. The

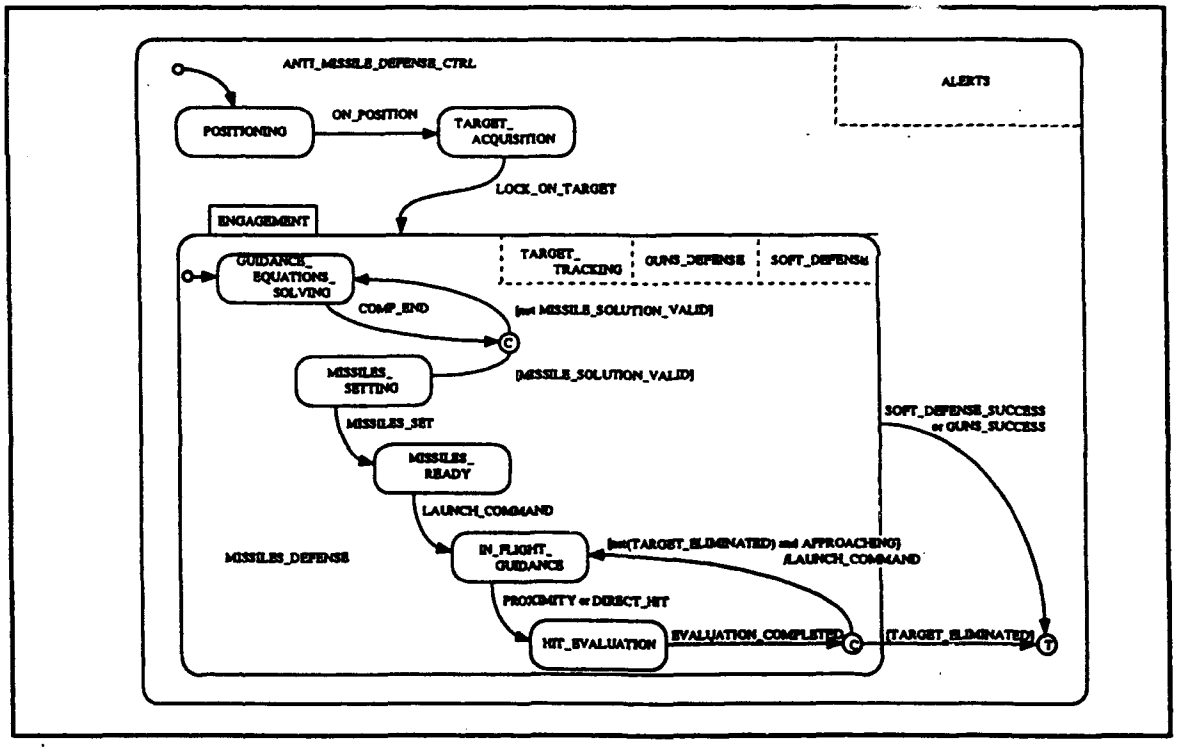


Figure 7 - MPDS ANTI_MISSILE_DEFENSE Process Control

State	Capability	Sub-system
Positioning	Position_FCS	WCS
Target_Acquisition	Acquire_Lock_Target	WCS
Soft_Defense	Deceive_Agressor	EW
	Check_Deception_Success	EW
Guidance_Equations_Solving	Solve_GDNC_Equations	WCS
Missiles_Setting	Prepare_Missiles	PDM
Missiles_Ready	Pre_Launch_Check	WCS
	Pre_Launch_Check	PDM
In_Flight_Guiudance	Launch	PDM
	Guide_Missiles	WCS
Hit_Evaluation	Evaluate_Hit	WCS
. . .	. . .	. . .

Table 2 - Mapping of Anti Missile Defense Process Sample Capabilities to States

corresponding simplified process control diagram is described by the Statechart shown in Figure 7.

The functional capabilities of the subsystems activated in each of the process states, correlating the two charts, are listed in Table 2.

The dashed AND line in Figure 7 is used to denote the concurrent execution of capabilities in a process. For example, the Target_Tracking capability of the WCS, Soft_defense capabilities of the EW, and defense guns preparation, pointing and firing capabilities of the PDG are performed concurrently with the activities performed in the MISSILES_DEFENSE state.

Transitions between states dictate the sequence of execution of the various activities of the sub-systems participating in the process.

Any possible thread (scenario) of the process resulting from a random combination of external (and internal) events can be identified and analyzed.

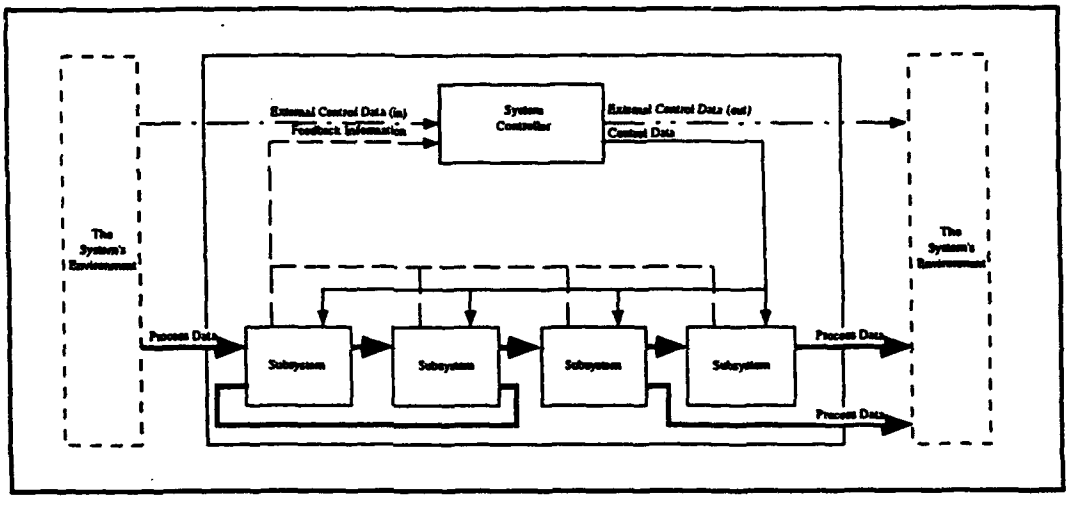


Figure 8 - ECSAM Computer Based System Generic Model

The Relationships Between the Conceptual Views

We will first briefly describe the ECSAM generic model of computer based systems, and building upon it we will explain the relationships between the conceptual views.

We assume that every computer based system and all of its subsystems can be modeled as a hierarchical control system according to the generic model shown in Figure 8 [LAVI84].

According to this model each system is decomposed into a set of logical subsystems and a conceptual controller which controls the system's joint operational behavior. The conceptual controller of the system presents the control of the operating modes and the control of the dynamic processes discussed earlier.

Logical subsystems and their conceptual controller can be further decomposed into logical subsystems and an

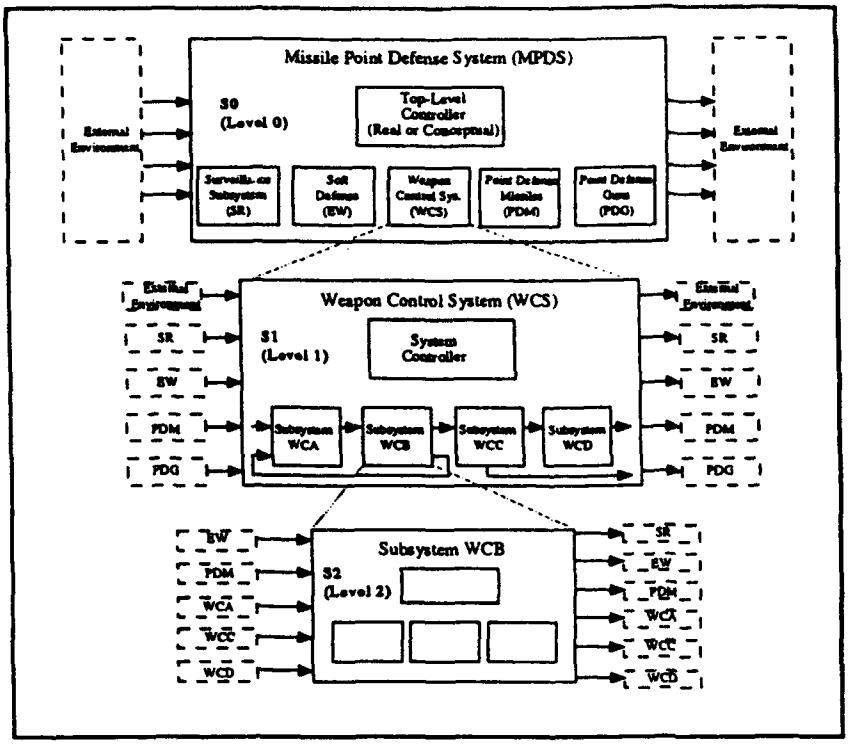


Figure 9 - ECSAM Description of Multi-level Systems

internal controller as shown in Figure 9. Naturally, each of the subsystems is also analyzed using the three conceptual views discussed earlier. The conceptual system controller (or control subsystem) controls the joint behavior of the subsystems' functional capabilities by controlling the system's operational modes and the dynamics of the system processes. Its functionality is thus defined by the system modes Statechart, and by the Statecharts of all system dynamic processes. The conceptual controller may include additional functions, e.g., the handling of the control signals affecting the dynamic transitions and redundancy management of critical systems.

Processes can be active throughout a mode or be activated by events while being in a mode, or can be periodically activated. The controller's implementation issues are treated during the system's design. It should be stressed that the services of the conceptual system controller can be implemented by a centralized controller or be distributed among many subsystems.

The relationships between the views is formalized in Figure 10. Figure 10-a shows the decomposition of a system into its logical subsystems. The functional capabilities performed by each subsystem are listed in Table a in Figure 10. The system's overall capabilities are shown in Figure 10-b. Their activation depends on the mode the system is in. These modes are shown in Figure 10-c. Each of the system capabilities P(n) is analyzed as a process described by the two diagrams shown in Figures 10-d and 10-e, where the components of the process are members of the set listed in table 10-a. This is shown in Figure 10-d where each process component is related to the subsystem executing it, using the notation M(k)>C(k,i), i.e., capability C(k,i) is performed by logical subsystem k. The relationships are valid for each logical system or subsystem at each level of the model in Figure 9. The analysis is performed iteratively at each level of the model.

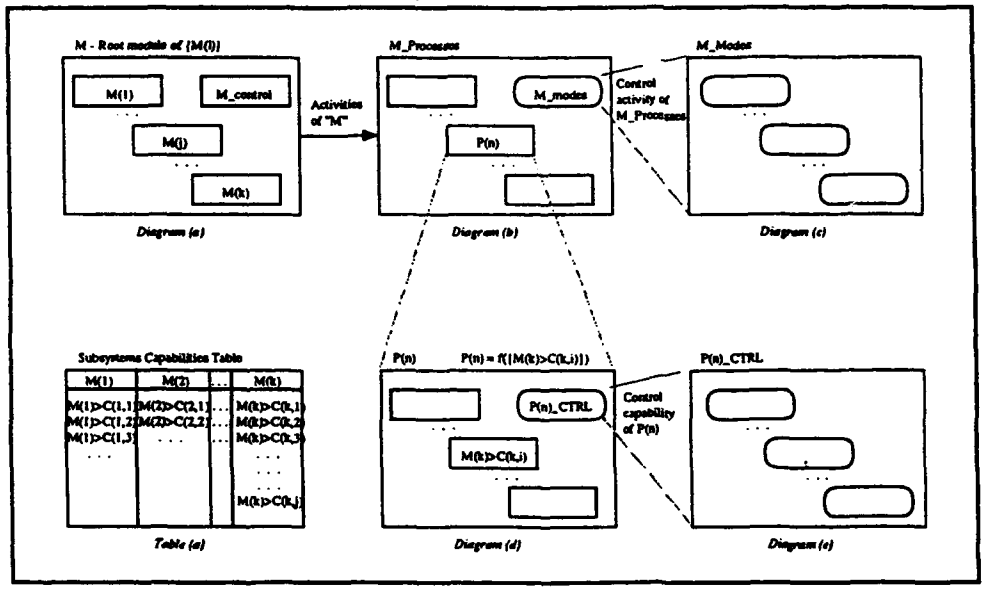


Figure 10 - Relationships Between ECSAM Conceptual Model Components

It is important to stress that the nature and number of system processes can change throughout the system's life cycle. However, once we have properly decomposed the system into logical subsystems, defined and implemented their functional capabilities, we can always program the system control to execute new or modified processes using the basic subsystems' capabilities implemented by the system.

External Specifications

Any system can be viewed as a part of a higher-level system. Practical systems can be

embedded in various environments, such as their operational environment, maintenance environment, testing environment, etc.

Although the information flows between the system and its environment may be similar (such as in the case when a system is operated in its operational environment and in a simulated one), the environment systems may differ.

ECSAM provides for the specification and analysis of external systems containing the System Under Development. In such cases, the SUD may be viewed as a "plug-in" component used in various applications.

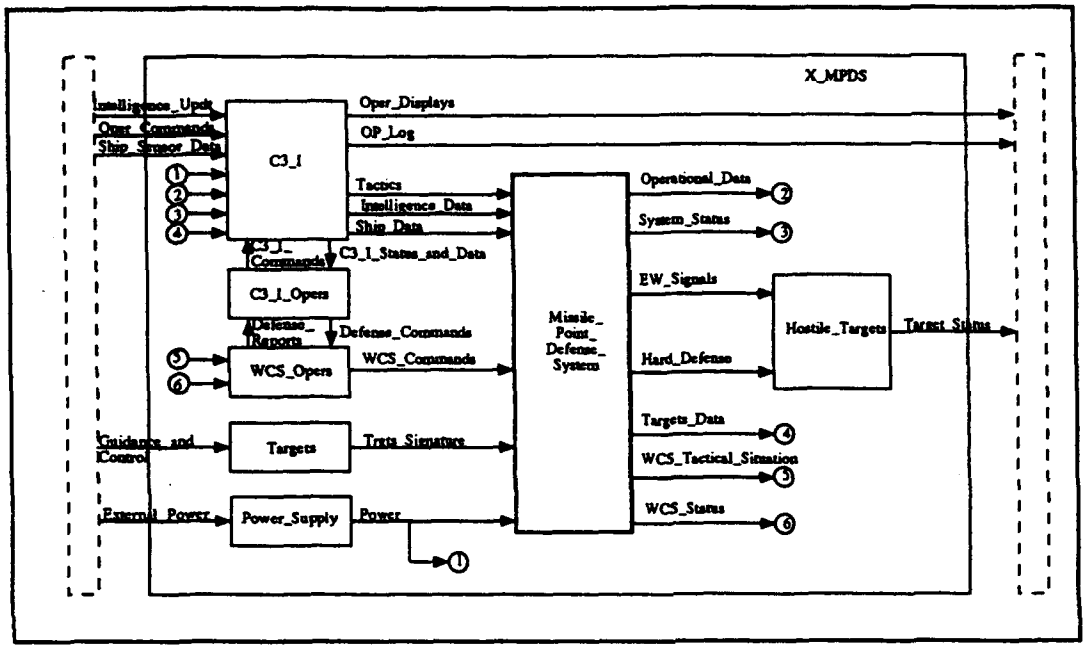


Figure 11 - The SUD as Part of the Overall System

Several different models describing the system in various environments may be developed for a SUD, e.g. for the testing and integration phase, for the operation phase and for the maintenance phase. During each of these phases the system may be operated in a different environment and consequently the specifications of the external behavior may be different. In Figure 11, the MPDS is portrayed in its operational environment. Other scenarios may require the definition of a different, suitable environment, as described above. The analysis of the SUD in various environments can contribute considerably to the quality of the design and the analysis of the SUD. The recursive nature of the ECSAM modeling and analysis methods allows the use of the very same generic models, semantics and methods in each level of a multilevel system analysis irrespective of the analysis level: the entire system or any of its lower-level subsystems.

4. The ECSAM Design Model

As described in previous sections, the conceptual model helps us to express the SUD's functionality, and serves as a vehicle for the analysis of the system's behavior and performance. The design model describes the system's architecture, the structure of its hardware and software subsystems, and their relationship. It also expresses the performance and implementation constraints, and describes the human - machine interface.

ECSAM provides a set of rules for the allocation of capabilities or complete logical modules, identified within a conceptual model, to a design model. Such a precise mapping is valuable, since it provides traceability from requirements to design, and from there - to implementation.

Typically, several different design models can be devised for any given conceptual model. This characteristic provides a foundation for the reuse of generic conceptual models in diverse implementations [LAV922]. The transition from a conceptual model to actual design necessitates the addition of various subsystems not expressed in the conceptual model (e.g., interfaces, communication facilities). These additional subsystems have conceptual models of their own. A design model usually expresses characteristics originating in several conceptual models.

ECSAM's design models have been described by us in detail in other papers [KUDI92, LAV921].

5. The Analysis and Design Process

Years-long engineering experience shows that the analysis process has to be iterative, and that several iterations are typically needed to produce a good final set of requirement specifications, a conceptual model, and an architectural view of the design model. The ECSAM method supports such rapid iterative processes.

ECSAM mandates the analysis of each layer or level of the system's conceptual and architectural models, and of each of its subsystems. Omitting the analysis of any view in any subsystem will result in incomplete (and typically even incorrect) specifications, in designs that do

not match the conceptual model, and in systems that do not fully answer the requirements. At IAI, all information generated during the iterative ECSAM process is continuously documented and cross checked using STATEMATE. Brief intermediate documents, which evolve over the project's life-cycle, are automatically generated to support the iterative process and its frequent internal reviews.

The basic ECSAM method analysis steps are outlined below. These steps are applied iteratively in the analysis at the system and subsystem levels. A full description of the process can be found in a separate report [LAV921]. This analysis process, proven in industrial projects, is also regularly taught in ECSAM courses at the IAI.

Description of the ECSAM Analysis and Design Process

The ECSAM analysis and design process consists of 13 main steps:

1. Definition of the system's scope and external requirements

   1.1 Definition of the system's scope

   The first step of the analysis is the definition of the basic scope of the system in terms of its main functions and basic performance criteria.

   1.2 Processing of customer requirements

   Concurrently with the definition of the system's scope and the analysis of the system in its environment (step 2), requirements are elicited from various applicable documents. These requirements are allocated to system models' components after completion of step 3, and they provide a foundation for the design and testing of the system. Requirements management and model development are carried out iteratively throughout the development cycle.

2. Definition of the system in its environment

   2.1 Preparation of the "System in its Environment" diagram

   The boundaries of the SUD are defined, and environment systems, inputs, and outputs are identified and described in a diagram.

   2.2 Preparation of short descriptions of the environmental systems and flows

   The environment systems are briefly described, and so are the information flows between the system and its environment.

3. Analysis of the top level logical modules, internal information flows, and capabilities

   3.1 Identification of logical subsystems

   Given the scope of the system, a basic set of requirements, and the description of the system in its environment, it is possible to identify the top level logical subsystems, and their major capabilities.

   The initial system decomposition is based on domain knowledge and on previous knowledge and experience gained in the development of similar systems and criteria, outlined in other papers [LAVI84, LAVI89].

   3.2 Analysis of major information flows

   Major system outputs and major system inputs (needed to produce the major outputs) are identified. Next, the paths of information flows are traced through the previously specified subsystems or modules going backward from the major outputs to the inputs.

   3.3 Identification of the subsystems' capabilities

   Each of the logical subsystems is next drawn with its environment on a separate chart. The logical subsystem's outputs are checked to verify the ability to produce them, considering the subsystem's inputs and an initial set of capabilities for each subsystem defined.

4. Initial definition of the system's basic operating modes

   The basic operating modes of the system are identified, and presented using the Statecharts notation. Valid transitions, and the events and conditions triggering them are not addressed at this step.

5. Definition of the architectural view of the design model

   The development of the architectural view of the design model requires several steps. Initially, the top level systems architecture is defined. At this phase the design involves the main architectural modules and their interconnections. To do so, issues such as the architectural component distribution, coupling, external and internal communications, redundancies and HMI concepts have to be resolved, based on a chosen design policy.

6. Identification of the system's external specification

   The external specification of the system relates to the operational requirements of the system. It is usually analyzed jointly by the operators' teams of the user/customer and the developers.

7. Preparation of the initial top level system specification

   Having completed the above steps, an initial top level specification document, which includes definition of the basic physical static and dynamic properties of the system is generated.

Questions or comments on content should be directed to:

Dr. Jonah Z. Lavi
CBSE Associates
3 Tabenkin St.
Ramat-Gan, Israel 52 302
lavi@math.tau.ac.il
972-3-571-8704

Or to:

Grady Campbell
Software Productivity Consortium
2214 Rock Hill Road
Herndon, VA 22070
campbelg@software.org
(703) 742-7104

Send feedback on the Consortium's Video Program and orders for video products to:

Technology Transfer Clearinghouse
Software Productivity Consortium
2214 Rock Hill Road
Herndon, VA 22070
brewer@software.org
(703) 742-7211